REPORT DOCUMENTATION PAGE

OMB No. 0704-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	Feb 19, 2001	FINAL REPORT 26Sep94-25Sep97

4. TITLE AND SUBTITLE

Development of a Multi-GHz Sampling Capability at the University of Maine UHF Test Facility

5. FUNDING NUMBERS

ARO Proposal Number 33635-RT-DPS

Contract/Grant Number DAAH04-94-G-0387

6. AUTHOR(S)

Donald M. Hummels
Fred Irons

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

University of Maine
5708 Barrows Hall
Orono, ME 04469-5708

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

U.S. Army Research Office
P.O. Box 12211
Research Triangle Park, NC 27709-2211

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

ARO33635.5-RT-DPS

11. SUPPLEMENTARY NOTES

The views, opinions and/or findings contained in this report are those of the author(s) and should not be construed as an official Department of the Army position, policy, or decision, unless so designated by other documentation.

12a. DISTRIBUTION/AVAILABILITY STATEMENT

Approved for public release; distribution unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

Abstract

The primary goal is to upgrade an existing 200 MHz high-speed ADC test facility to obtain 20 GHz test and measurement capability. The modification is required to keep abreast of developing technology in ADC design, and in particular to support an ongoing effort in the ARPA HBT/ADC program to develop multi-GHz analog-to-digital converters. Also, innovative test methodologies are being developed to characterize and diagnose distortion mechanisms for state-of-the-art converters with sample rates above 1 GHz.

An important accomplishment is the development of diagnostic test procedures which may be used on fully packaged components. Normally it is not practical to probe internal points in high-speed circuits – due to loading and unloading transmission line effects – and so external diagnostic procedures are very desirable. Under this contract we have shown how phase-plane error functions can be built and interpreted to estimate specific ADC architecture effects. Additionally, this work has shown that the same set of calibration data can be used to estimate different features through the appropriate choices of basis functions related to specific ADC architectures. Fast Orthogonal Search methods were developed to assist in the selection of the most sensitive error basis functions.

14. SUBJECT TERMS

15. NUMBER OF PAGES 77

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL

Final Report
Development of a Multi-GHz Sampling Capability at the University of Maine UHF Test Facility

Contractor Name: University of Maine

Principal Investigators:

Donald M. Hummels
Principal Investigator
5708 Barrows Hall
Orono, ME 04469-5708
(207) 581-2245
(207) 581-2237 (LAB)
hummels@eece.maine.edu

Fred Irons
Co-Principal Investigator
5708 Barrows Hall
Orono, ME 04469-5708
(207) 581-2229
(207) 581-2237 (LAB)
irons@eece.maine.edu

Contract Number: DAAH04-94-G-0387
ARO Number: 33635-RT-DPS

Abstract

The primary goal is to upgrade an existing 200 MHz high-speed ADC test facility to obtain 20 GHz test and measurement capability. The modification is required to keep abreast of developing technology in ADC design, and in particular to support an ongoing effort in the ARPA HBT/ADC program to develop multi-GHz analog-to-digital converters. Also, innovative test methodologies are being developed to characterize and diagnose distortion mechanisms for state-of-the-art converters with sample rates above 1 GHz.

An important accomplishment is the development of diagnostic test procedures which may be used on fully packaged components. Normally it is not practical to probe internal points in high-speed circuits – due to loading and unloading transmission line effects – and so external diagnostic procedures are very desirable. Under this contract we have shown how phase-plane error functions can be built and interpreted to estimate specific ADC architecture effects. Additionally, this work has shown that the same set of calibration data can be used to estimate different features through the appropriate choices of basis functions related to specific ADC architectures. Fast Orthogonal Search methods were developed to assist in the selection of the most sensitive error basis functions.

1 Statement of the Problem Studied

The primary goal is to upgrade an existing 200 MHz high-speed ADC test facility to obtain 20 GHz test and measurement capability. The bulk of the project cost is dedicated to equipment purchases to achieve this upgrade. The modification is required to keep abreast of developing technology in ADC design, and in particular to support an ongoing effort in the ARPA HBT/ADC program to develop multi-GHz analog-to-digital converters.

In addition, this project extends the University of Maine's involvement in test support for the ARPA HBT/ADC program. Under this program, innovative test methodologies were developed to characterize and diagnose distortion mechanisms for state-of-the-art converters with sample rates above 1 GHz.

2 Summary of Important Results

The following items summarize specific accomplishments of the research effort. The primary accomplishments are the development of the high-speed test capability and the development of diagnostic test procedures which may be used on fully packaged components. Sophisticated test control software has been developed to allow access to the test facility from remote locations. Normally it is not practical to probe internal points in high-speed circuits – due to loading and unloading transmission line effects – and so external diagnostic procedures are very desirable. Under this contract we have shown how phase-plane error functions can be built and interpreted to estimate specific ADC architecture effects. Additionally, this work has shown that the same set of calibration data can be used to estimate different features through the appropriate choices of basis functions related to specific ADC architectures. Fast Orthogonal Search methods were developed to assist in the selection of the most sensitive error basis functions, and histogram test techniques have been developed to improve the resolution of previous calibration schemes.

Important results of the research include the following items:

- The proposed modifications to the University of Maine test facility were made. The following equipment was purchased and added to the U-Maine high-speed ADC test facility:

HP83712A	Synthesized CW Generator (0.01-20 GHz)
HP83732A	Synthesized CW Generator (0.01-20 GHz)
HP70000	Spectrum Analyzer (100 Hz - 26.5 GHz)
Wil 37247A	Vector Network Analyzer (0.04-20 GHz)
HP16500B	Logic Analysis System (32 bits at 1 Gsps)

  In addition, the University has supplied two DEC Alpha workstations which are used exclusively for test-facility support and GPIB control of the testbed.

- Demonstrated that phase-plane error functions are valid for all ADCs produced using the same design and IC masks. This was demonstrated on commercial high-speed components for units produced more than one year apart.

- Published a time-domain diagnostic procedure for digital data acquisition.
- Developed techniques to interpret ADC error functions for the estimation of specific ADC architectural errors. Improved orthogonal search techniques were developed to identify dominant sources of error. Emphasis was placed on the Folding and Interpolating ADC architectures being developed under the ARPA program. For this architecture, diagnosable error mechanisms include: comparator hysteresis, integral nonlinearities, state-dependent sample-time jitter, and threshold-voltage reference resistor errors.
- Developed and published error correction techniques for time-interleaving sampling structures.
- Developed and published methods of estimating random timing jitter components through a statistical analysis of residual error.
- Analyzed raw test data supplied by Lincoln Laboratory for both TRW and Rockwell prototypes as well as probe data supplied by Rockwell.
- Developed and published custom networked software for control of ADC test equipment.
- Improved phase plane error description techniques using histograms. The improvements allow differential ADC nonlinearities to be incorporated into the above techniques.
- Developed error models for Sigma-Delta oversampled converters. Identified test techniques to compensate for dominant distortions sources.

3 Publications and Technical Reports

The following publications were funded in part by this project:

- J. Larrabee, F. Irons, and D. Hummels, "Using sine wave histograms to estimate analog-to-digital converter dynamic error functions," *IEEE Trans. Instrumentation and Measurement*, vol. 47, pp. 1448–1456, Dec. 1998.
- D. Rawnsley, D. Hummels, , and B. Segee, "A virtual instrument bus using network programming," in *Proc. of the ASEE Annual Conference*, (Milwaukee, WI), June 1997.
- D. Hummels, D. Gerow, , and F. Irons, "A compensation technique for sigma-delta analog-to-digital converters," in *Proceedings of IEEE International Instrumentation and Measurement Technology Conference*, (Ottawa Canada), pp. 1309–1312, May 1997.
- J. Larrabee, D. Hummels, , and F. Irons, "ADC compensation using sinewave histogram methods," in *Proceedings of IEEE International Instrumentation and Measurement Technology Conference*, (Ottawa Canada), pp. 628–631, May 1997.
- D. Rawnsley, D. Hummels, , and F. Irons, "A virtual instrument bus using network programming," in *Proceedings of IEEE International Instrumentation and Measurement Technology Conference*, (Ottawa Canada), pp. 694–697, May 1997.
- F. Irons and D. Hummels, "The modulo-time plot – a useful data acquisition diagnostic tool," *IEEE Trans. Instrumentation and Measurement*, vol. 45, pp. 734–738, June 1996.
- F. Irons, D. Hummels, and I. Papantonopoulos, "ADC error diagnosis," in *Proceedings of IEEE International Instrumentation and Measurement Technology Conference*, (Brussels), pp. 732–737, June 1996.
- D. Hummels, J. McDonald, and F. Irons, "Distortion compensation for time-interleaved ADCs," in *Proceedings of IEEE International Instrumentation and Measurement Technology Conference*, (Brussels), pp. 728–731, June 1996.
- D. Hummels, I. Papanonopoulos, and F. Irons, "Identification of error mechanisms in a folding and interpolating ADC," in *Proceedings of IEEE International Symp. on Circuits and Systems*, (Atlanta, GA), pp. 176–179, May 1996.
- F. Irons, D. Hummels, and C. Zoldi, "ADC architectural diagnostic testing procedures," in *Proceedings of Government MicroCircuit Applications Conference*, (Orlando), pp. 79–80, Mar. 1996.
- D. Hummels, W. Ahmed, and F. Irons, "Measurement of random sample time jitter for ADCs," in *Proceedings of IEEE International Symp. on Circuits and Systems*, (Seattle), pp. 708–711, May 1995.
- W. Ahmed, D. Hummels, and M. Musavi, "Application of fast orthogonal search for the design of RBFNN," in *Proceedings of IEEE International Symp. on Circuits and Systems*, (Seattle), pp. 1952–1955, May 1995.

In addition, the following University of Maine Masters Theses were funded in part by this project:

- Jon Larrabee, August 1997, "ADC Compensation Using a Sinewave Histogram Method".
- Daryl Rawnsley, May 1997, "Virtual Instrument Bus Using Network Programming".
- Deron Gerow, August 1996, "Error Mechanisms In Sigma-Delta Analog-to-Digital Converters".
- Ioannis Papantonopoulos, August 1995, "Error Modeling for Folding and Interpolating Analog to Digital Converters".
- Wahid Ahmed, August 1994, "Fast Orthogonal Search for Training Radial Basis Function Neural Networks".

4 Scientific Personnel

Faculty:	Donald M. Hummels and Fred H. Irons
Graduate Students:	Jon Larrabee, MS EE August 1997 Daryl Rawnsley, MS EE May 1997 Deron Gerow, MS EE August 1996 Ioannis Papantonopoulos, MS EE August 1995 Wahid Ahmed, MS EE August 1994

5 Report of Inventions

None.

Copies of Publications

Using Sine Wave Histograms to Estimate Analog-to-Digital Converter Dynamic Error Functions

J. Larrabee, F.H.Irons, and D.M.Hummels*

February 16, 2001

*This work has been supported in part by the ARPA HBT/ADC program under a contract administered by the Office of Naval Research Grant N000149311007 and the DEPSCor program through the Army Research Office Grant DAAH04-94-G-0387

Abstract

This paper describes a new method for developing analog-to-digital converter (ADC) error function models using modified sinewave histogram methods. The error models may be used to digitally compensate for nonlinearities introduced by the converter. The histogram modification involves sorting of converter output samples based upon an estimated associated input derivative signal. This error model is based upon a previously unpublished result which shows that sinewave histograms yield distinctly different expected errors for each state based upon input signal slope associated with each output sample. This result thus provides a dynamic dependence for expected errors measured by means of histogram methods.

Sorted sinewave histograms are used to estimate slope dependent expected errors at each ADC output state (code). The method provides improved error representation by providing error basis functions for every output code. Simulated results prove that this method removes all slope dependent errors for complex ADC architectures while experimental results for an 8-bit 200 MSPS ADC yielded more than 10 dB improvement in spurious-free-dynamic-range (SFDR) over the full Nyquist band. The new method is thus shown to possess wideband dynamic error character.

1 Introduction

This paper presents a complete development for the representation of ADC dynamic error as a function of output state (ADC output code) and input signal slope, through the use of modified histograms[1]. Modeling ADC error as a function of state and slope of the input signal is a concept that has been under development for some time [2, 3, 4]. Previous calibration schemes [5, 4] used smooth basis functions, over output state and input signal slope, to model ADC error. The ADC is driven with a single tone sinusoid and an FFT spectrum is obtained from sample sets acquired from the converter's output. The harmonic components of the spectrum that are above the noise floor are considered to be ADC error and are used to calculate an ADC error model. The predicted error from this model may be stored in a look-up table to be used for compensating converter error.

A completely different method for estimating an ADC *dynamic* error function is the basis for this paper. The method is based upon *expected* error for each state of the converter as a function of the slope of the input to the ADC. Modified histogram techniques are used to obtain estimates of expected error which in turn provide least square curve fits of error-versus-slope for *each state.* It is known that ADCs contain discontinuous expected error from state-to-state and these discontinuities can not be accurately described when error is averaged over a range of states as in previous compensation methods [2, 3, 4]. The procedure developed in this paper finds a smooth error function in slope for each state. By increasing resolution in the state domain, discontinuities in error are accurately modeled from state-to-state, thus providing improved ADC error representation.

Section 2 describes and develops the ADC error model which is required to implement the desired dynamic error characterization. Histograms, obtained from ADC response to pure sinewaves, are used to estimate quantization thresholds which in turn are used to estimate expected error for each ADC state. A procedure is then developed that estimates error function basis coefficients for each state. It is shown, in Section 3, that two unknown parameters exist for each calibration signal, namely : the amplitude and DC offset of the input signal. A constrained least-squares approach is used to estimate each input parameter so that desired error function basis coefficients can be estimated. Section 4 concludes the paper by providing examples of the compensation of raw ADC data using the error model developed. Effectiveness of the error model is first evaluated through the use of a simulated ADC performance to show that the proposed method compensates for all output code and input slope dependent errors. The error compensation method is then applied to a real wideband 8-bit flash ADC where it is found that dynamic performance is improved by as much as 10 dB over the full Nyquist band of the ADC.

[1]The paper is a follow up to the paper presented at the International Conference in Ottawa [1] and provides all details required to implement the algorithm.

2 ADC Error Modeling

In theory, analog-to-digital conversion transforms continuous time signals into signals that exist on equally spaced time intervals at discrete output values. When the conversion is ideal, the only error introduced by this process is quantization error from the digitization of the analog signal. By using a dithered input signal, quantization error is made to appear as additive noise, with a resultant noise level that depends upon the resolution of the ADC [6]. However, most conversion is not ideal and errors are introduced from ADC architectural non-idealities and mismatched component values. This section develops a generic state and slope dependent error model for non-ideal ADC performance. A standard static error function, obtained from a sinewave histogram, is modified by sorting ADC sample sets into two halves corresponding to positive and negative values of the input signal derivative (slope).

2.1 An ADC Transfer Function Model

ADC performance can be described in terms of the transfer function of output code versus input voltage [7]. Each ADC output code, i, is associated with two input quantization thresholds, t_i and t_{i+1}. These quantization thresholds describe the range of input voltage that yields the ith output code. By measuring all output codes versus their input thresholds, the ADC transfer function is created. For an ideal converter, the quantization thresholds are equally spaced, so that the output code is proportional to the input voltage. Non-ideal quantizers display a non-linear relationship between input voltage and output code as an approximation to a straight line. A "Nominal Characteristic" is defined to be a straight line to which a given quantizer approximates. This characteristic is represented as a line with slope G and intercept x_o.

Expected error, a common measure of ADC quality, is defined to be, at the ith state, the average deviation of the actual transfer function from the nominal characteristic. Fig. 1 shows the ADC error, $e(x)$, as a function of input voltage, x, and it is constructed by subtracting the nominal characteristic, $G(x - x_o)$, from the actual transfer function and zooming in on the ith output state. The average error for state i is referred to as *expected error* and is denoted by E_i. The expected error for state i is written in terms of quantization thresholds t_i by finding the midpoint of the error curve shown in Fig. 1.

$$E_i = i - G\left(\frac{t_i + t_{i+1}}{2} - x_o\right) \qquad i = 1, 2, \ldots, 2^n - 2 \tag{1}$$

G and x_o are the gain and offset of the nominal characteristic, n is the number of bits of the ADC, and t_i, t_{i+1} are quantization thresholds that bracket the ith state. Equation (1) gives the expected error as a function of output code in terms of the quantization thresholds t_i and t_{i+1}. A method for estimating quantization thresholds is now developed in order to use (1).

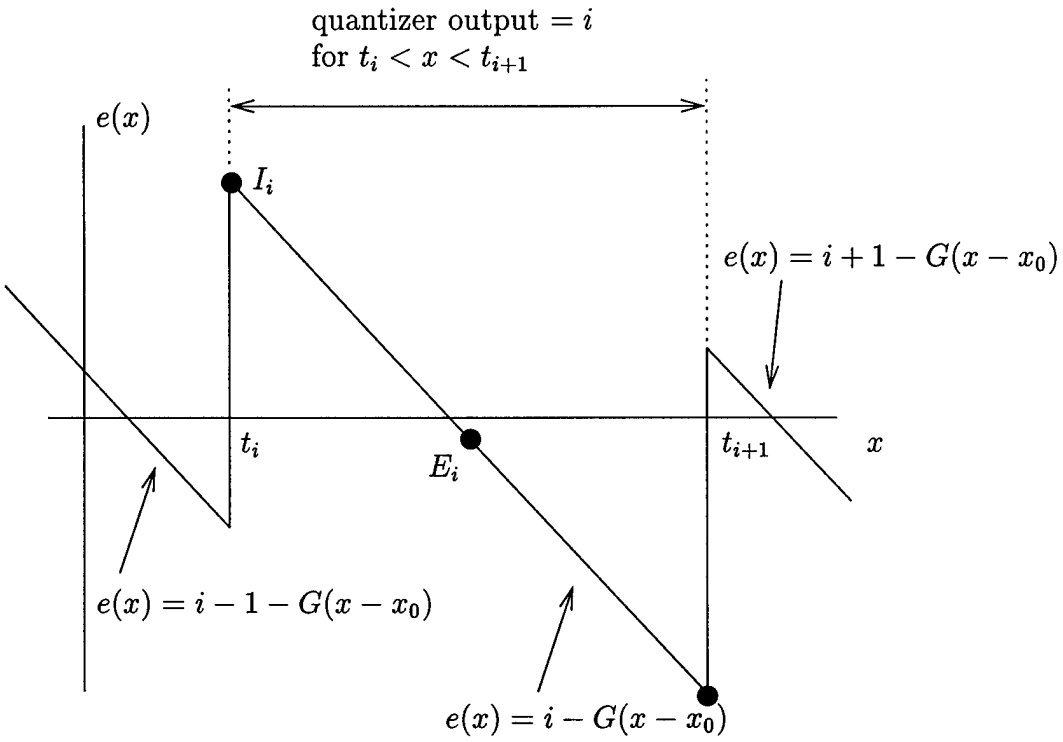

Figure 1: Static error description

2.2 Histogram Methods

Histogram tests offer a way to estimate quantization thresholds which are necessary in order to estimate the expected error of (1). Creating a histogram over an output sample set involves counting how many times each output code occurs. A relationship for state i between the number of occurrences relative to the total number of samples in the set and the quantization interval, $t_{i+1} - t_i$, is established. For example, if a ramp input is applied to an ADC, and a large number of samples is collected, then ideally a histogram should contain the same number of occurrences in each state. When one state contains fewer occurrences than the average, its quantization width is correspondingly smaller than the average width. The opposite is true when a state's number of occurrences is larger than the average. Accurate estimates of quantization thresholds are determined whenever enough samples are obtained.

A formal procedure for estimating thresholds can be developed when the input signal to the ADC is known. Let the values (voltages) of an input sinusoid be denoted by the random variable X. Its probability distribution function is shown in Fig. 2 for a case in which X is a random sample of a sinusoid with amplitude A and DC offset c. Now let Y represent the random variable of a non-ideal quantizer output. Y takes on integer values of $0, 1, 2, \ldots, 2^n - 1$ and its distribution function may be estimated from a histogram taken on an output sample set. Given a set of N samples, the estimated distribution function is given by $\hat{F}_Y(i) = S_i/N$, where S_i is the cumulative number of occurrences for output codes 0 through i.

The distribution functions of X and Y are related to the values of the quantization thresholds, t_i, as shown in (2).

$$F_X(t_i) = P(X \leq t_i) = P(Y \leq i - 1) = F_Y(i - 1) \tag{2}$$

By combining the estimated distribution for Y and the known functional form for $F_X(x)$, (2) can be used to derive an estimate of t_i: $\hat{t}_i = F_X^{-1}(S_{i-1}/N)$. For a sinusoidal signal, $\hat{t}_i$ is obtained from (3).

$$F_X(\hat{t}_i) = \frac{1}{2} + \frac{1}{\pi} \arcsin\left(\frac{\hat{t}_i - c}{A}\right) = \frac{S_{i-1}}{N} \tag{3}$$

where $i = 1, 2, \ldots, 2^n - 1$. Solving (3) for $\hat{t}_i$ gives (4) which is the same relation given in the IEEE Waveform Recorder Standard, [7].

$$\hat{t}_i = -A \cos\left(\pi \frac{S_{i-1}}{N}\right) + c, \qquad i = 1, 2, \ldots, 2^n - 1 \tag{4}$$

A description for the estimated expected error of a particular ADC output code is obtained by combining (4) with (1) to obtain (5).

$$\hat{E}_i = i + \frac{GA}{2}\left(cos\left(\pi \frac{S_{i-1}}{N}\right) + cos\left(\pi \frac{S_i}{N}\right)\right) - G(c - x_o) \tag{5}$$

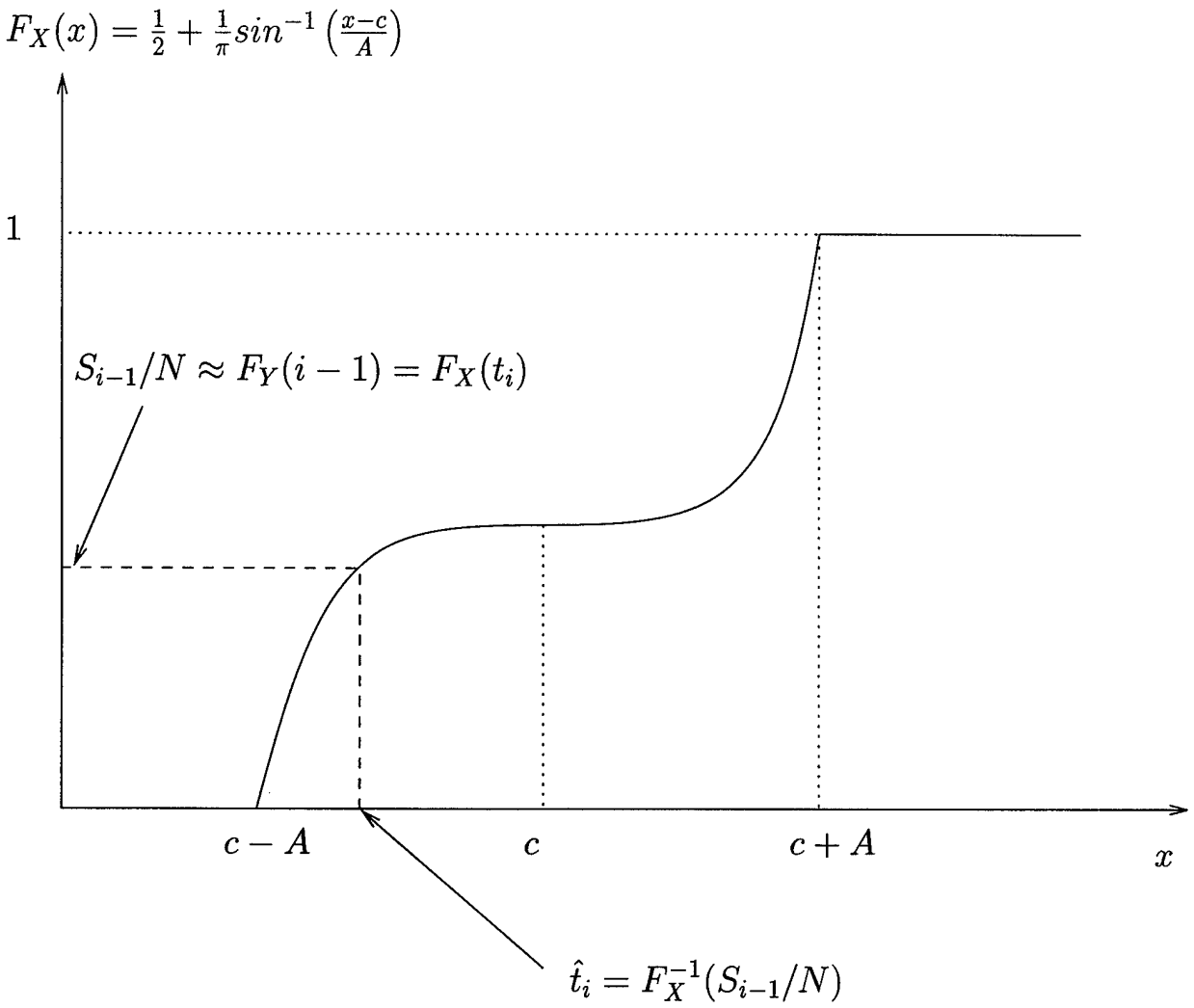

Figure 2: Distribution function for sinewave with amplitude A and offset c

where $i = 1, 2, \ldots, 2^n - 2$. Equation (5) contains two unknown parameters, $p_0 = G(c - x_0)$ and $p_1 = GA$. The true amplitude, A, and offset, c, of an input sinusoid is difficult to control and know with sufficient accuracy. Both the converter circuitry and the test setup introduce gain and offset errors to the input signal of the ADC.

For a single test sinusoid, the parameters A and c can be estimated from an FFT of the sample set. Two common methods are used to obtain reasonable values for the characteristic parameters of an ADC transfer function [7]. The "independent-based" method chooses G and x_o to minimize the sum-squared expected error over all states. The "terminal-based" approach selects G and x_o so as to zero the integral error (peak of the quantizer error function) at t_1 and t_{2^n-1} to obtain (6).

$$\begin{aligned} G &= \frac{2^n - 2}{t_{2^n-1} - t_1} \\ x_o &= t_1 - 1/G \end{aligned} \tag{6}$$

G and x_o are actually global parameters so that when dealing with several sets of calibration data, neither method is consistent when used independently on each sinusoid due to the fact that each sample set yields different values for G and x_o. How to deal with this problem is a prime difficulty with the formulation of the proposed method. The development in Section 3 solves this problem through a constrained least squares formulation.

However, an important characteristic of expected dynamic errors for an ADC must be discussed first. For practical converters, the expected error for an ADC depends on the dynamic behavior of the input signal. Fig. 3 shows estimated expected errors for the full range of a Tektronix AD20 8-bit ADC for a single sinusoidal input. The samples were divided into two sets based upon the polarity of the slope of the sampled signal. The sorting is achieved by obtaining a slope estimate associated with each sample and separating the samples into positive and negative slope sets. The terminal-based approach [7] is used to estimate parameters, G and x_o, and (5) is used to obtain expected errors for each sample set. From Fig. 3 it is evident that estimated expected error follows a different pattern for the positive slope set than it does for the negative slope set. Thus, the error model described by (5) is incomplete since it has no associated input signal slope dependency. The following section deals with this deficiency. Two additional features should be noted in connection with Fig. 3. First, the curves are repeatable for a good ADC in both the regular smooth pattern and the smaller irregular state-to-state errors. In addition, the regular pattern is frequency-dependent and thus cannot be removed with a static calibration function of state only. The repeatable part of the irregular error provides the basis for the state and slope dependent error function developed in this paper.

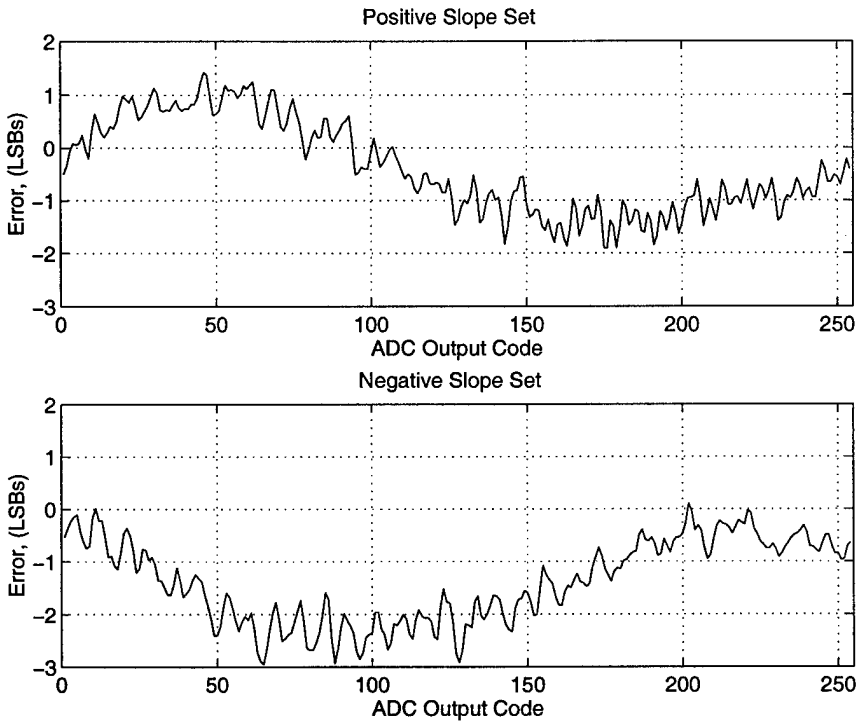

Figure 3: Estimated expected error for $\pm$ slope sample sets

2.3 Error Model as a Function of State and Slope

As pointed out above, estimated expected error not only depends on the value of the output state, i, but it also depends upon the slope of the input signal, y. In this paper, ADC error is modeled by a separate error function, $e_i(y)$, for each converter output code. This model is a departure from previous ADC error models [8, 9] which employed smooth functions of both output code and input slope. By using separate error functions for each output code, discontinuous error functions reflecting differential state-to-state ADC errors are developed as required for most ADC architectures. Consequently, to represent a desired error function, each expected output error is written in (7) as a linear combination of a set of basis functions.

$$\hat{e}_i(y) = \sum_{k=1}^{M} \alpha_{i,k} b_k(y) \tag{7}$$

$\hat{e}_i(y)$ is estimated expected error at state i evaluated at slope y, $b_k(y)$ is the k_{th} basis function evaluated at slope y, and $\alpha_{i,k}$ is the k_{th} basis function coefficient for state i. Equation (7) yields an error function of slope for $2^n - 2$ states since the expected error is only defined for each bounded quantization interval of the converter. The data of Fig. 3 show that practical converters exhibit slope dependent discontinuous error-versus-slope patterns between adjacent output codes. Allowing separate error-versus-slope functions for every state increases the effectiveness of these error models relative to previous methods. The next section develops procedures required to estimate the combined parameters, p_0 and p_1, of (5) and the basis function coefficients, $\alpha_{i,k}$, of (7) for each state and sample data set.

3 Calibrating The ADC

Section 2 described an error model dependent on output code and input slope. This section describes how to use measured data to estimate model parameters. Data is collected from the ADC to accurately characterize the converter according to the proposed error model. Sinusoids are the chosen input signals since they provide dynamic behavior for use in the error model and are easily generated with sufficient purity. Once data are collected, they must be manipulated into a useful form for developing error basis function coefficients. The remainder of this section addresses the steps of collecting raw ADC sample sets and using the collected data to estimate ADC error as a function of state and slope.

Calibration of an ADC involves the selection of model parameters, $\alpha_{i,k}$ in (7), to predict its expected error. Sinewave histogram techniques are used to accomplish this task. To accurately describe error, the ADC must be excited over its full range of state and slope values. Error observed in response to applied test signals determines appropriate coefficients for the basis functions used in (7).

Each sinusoidal test signal generates an elliptical trajectory in state and slope space. Varying the amplitude and frequency of a sinusoid changes the shape of the ellipse. The amplitudes and frequencies of the sinusoids must be carefully selected for accurate ADC characterization. Large frequencies create greater ranges in slope values and large amplitudes excite more output codes. Choosing various frequencies and amplitudes yields a series of trajectories as shown in Fig. 4. The goal is to choose a series of sinusoids at varying amplitudes and frequencies that fill the state and slope space (the domain of the error model) while providing a sufficient number of intersections across each state so as to reduce ambiguity in the least-square estimation of the basis function coefficients. For example, in Fig. 4, states just less than 250 would have only 10 intersections and could only only solve uniquely for 10 or fewer basis coefficients in that region whereas states near 200 would have 30 intersections with trajectories and could support the estimate of up to 30 parameters.

3.1 Calibration Overview

A large number of samples are collected for each sinusoidal input to the ADC. Slope estimates are obtained from the samples and separate histograms are constructed based upon positive and negative slopes to form estimates of the expected error for the upper and lower half of each trajectory. In this paper, $16k$ samples are collected for each trajectory. This allows roughly $8k$ samples in each of the positive and negative slope sets. There are 256 output codes on an 8-bit ADC and so the $8k$ samples are spread over 256 bins when a fullscale histogram is constructed. There must be enough occurences, S_i, for each output code in order to obtain an accurate estimate of the quantization thresholds, $\hat{t}_i$.

A single trajectory yields two histograms, one each for positive sloped and negative sloped samples. Typically the slope is estimated by either using an

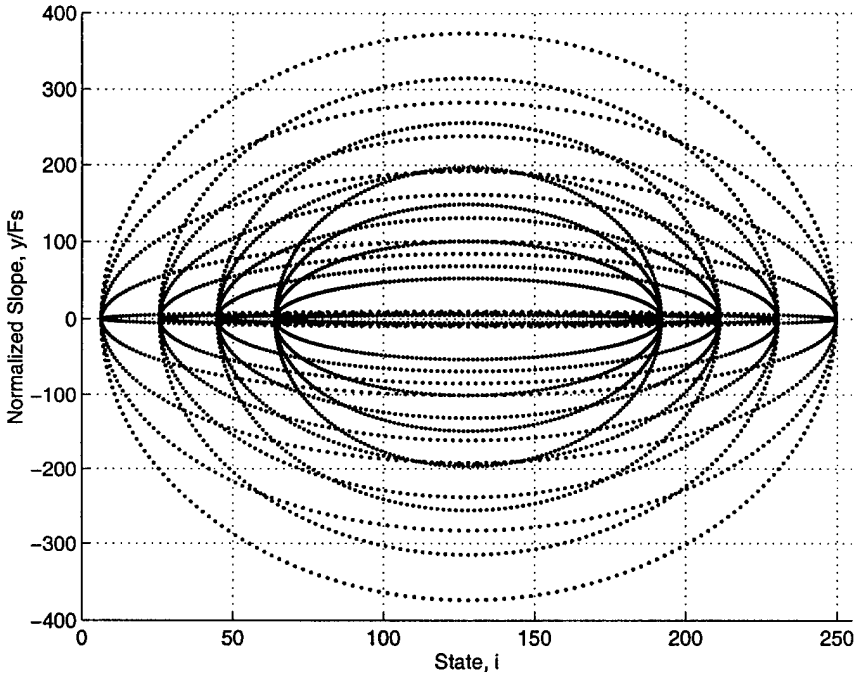

Figure 4: Trajectory plot using 5 frequencies at 4 amplitudes per frequency

FIR filter on the raw data or by $j\omega$-multiplication in the frequency domain of the transformed sample set. Equation (4) gives an expression for estimating thresholds based on the S_i sums. However, the amplitude and offset, A and c of the input sinusoid are usually unknown (and frequency dependent). Each histogram yields the measured S_i variables required in (5), however, the parameters, $p_1 = GA$, $p_0 = G(c - x_o)$, and $\alpha_{i,k}$, still need to be determined to complete the solution of the problem.

In implementation, significant memory savings are realized by reducing data prior to estimation of the basis coefficients. The goal is to construct a single error value at each state i and slope y, for (7). For example, a single trajectory of $16k$ samples yields two sample sets (positive and negative slopes) of roughly $8k$ samples per set which are used to construct two histograms of length 2^n, where n is the number of converter bits. Then at every output code i, there exist two slope estimates, y^+ and y^-, one for the positive and one for the negative slope half of each trajectory. The estimated slopes associated with each state i are averaged to yield slope estimates for each of the positive and negative slope sets.

A direct (but ineffective) way to estimate the gains and offset parameters is to use an FFT of a sample set to obtain estimates for c and A and then use a terminal or independent-based method [7] to estimate G and x_o. This procedure could be used individually for each trajectory and the resulting estimated parameters would then be used with histogram data to calculate expected errors. Every ADC output state i, excited by a trajectory, has two slope and expected error values associated with it. For state i, the basis functions of (7) could then be used to "best-fit" a curve versus slope to the expected errors from all trajectories. The result of this procedure gives a description of expected error based on state and slope of the input. However, experiments have shown that this method for finding c, A, G, and x_o individually for each trajectory causes a severe degradation in error model accuracy. The degradation is due to the fact that A and c represent the converter *input* amplitude and offset, and a frequency-dependent bias is introduced by estimating these terms from the converter *output*. A better approach must be used to estimate all unknown parameters for each trajectory.

3.2 A Consistent Formulation of the Error Model Analysis

The following discussion provides a mathematical description on how to estimate suitable basis function coefficients for the proposed error model.

Equation (5) is rewritten in parameterized form as follows.

$$\begin{aligned} \hat{E}_i &= i - p_1 h_i - p_0 \\ p_0 &= G(c - x_o), \;\; p_1 = GA \\ h_i &= -\left[cos(\pi S_{i-1}/N) + cos(\pi S_i/N)\right]/2 \end{aligned} \tag{8}$$

$\hat{E}_i$ is the estimated expected error for output code i, p_0 and p_1 are unknowns, and h_i is a known measurement obtained from a histogram. The next section

shows how to write the desired basis coefficients as a function of the parameters, p_0 and p_1.

3.2.1 Casting the Basis Coefficients in Terms of p_0 and p_1

From Section 2 an error model as a function of output state and input slope was developed in (7) as a dot product of basis functions with unknown basis coefficients, $\alpha_{i,k}$. A vector form of this model is given by (9) where $\vec{\alpha}_i$ is an M-element column vector of basis coefficients at the ith state and $\vec{b}^T(y)$ is an M-element row vector of basis functions evaluated at input slope, y.

$$\hat{e}_i(y) = \vec{b}^T(y)\vec{\alpha}_i \tag{9}$$

Setting both histogram estimated errors of the jth trajectory at state i equal to the modeled error of (7) provides a matrix relation as in (10).

$$\begin{bmatrix} \hat{E}_i^+ \\ \hat{E}_i^- \end{bmatrix} = i\begin{bmatrix} 1 \\ 1 \end{bmatrix} - \begin{bmatrix} 1 & h_i^{+jth} \\ 1 & h_i^{-jth} \end{bmatrix}\begin{bmatrix} p_0^{jth} \\ p_1^{jth} \end{bmatrix} \approx \begin{bmatrix} \vec{b}^T(y_i^{+jth}) \\ \vec{b}^T(y_i^{-jth}) \end{bmatrix}\vec{\alpha}_i \tag{10}$$

The $+jth$ and $-jth$ superscripts indicate respectively the positive and negative sloped portions of the jth trajectory. p_0^{jth} and p_1^{jth} are the unknown global parameters associated with the jth trajectory. These parameters are the same for both slope sets since the sets come from the same input. $\hat{E}_i^+$ and $\hat{E}_i^-$ are the estimated expected errors based on positive and negative sloped sample sets respectively. The problem is to solve for the basis function coefficients, $\vec{\alpha}_i$, $(i = 1, 2, \ldots, 2^n - 2)$, and the unknown trajectory parameters p_0^{jth} and p_1^{jth} which minimizes residual error in the approximation of (10). y_i^{+jth} and y_i^{-jth} are the average positive and negative slopes for output code i of the jth trajectory. h_i^{+jth} and h_i^{-jth} terms are constructed from the positive and negative slope histograms respectively. The relationship described by (10) is expanded over k_T trajectories that drive the ADC creating the vector relationship given in (11) for the ith output code.

$$i\begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ \vdots \\ 1 \\ 1 \end{bmatrix} - \begin{bmatrix} 1 & h_i^{+1st} & 0 & 0 & \ldots & 0 & 0 \\ 1 & h_i^{-1st} & 0 & 0 & \ldots & 0 & 0 \\ 0 & 0 & 1 & h_i^{+2nd} & \ldots & 0 & 0 \\ 0 & 0 & 1 & h_i^{-2nd} & \ldots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \ldots & 1 & h_i^{+k_Tth} \\ 0 & 0 & 0 & 0 & \ldots & 1 & h_i^{-k_Tth} \end{bmatrix}\begin{bmatrix} p_0^{1st} \\ p_1^{1st} \\ p_0^{2nd} \\ p_1^{2nd} \\ \vdots \\ p_0^{k_Tth} \\ p_1^{k_Tth} \end{bmatrix} \approx \begin{bmatrix} \vec{b}^T(y_i^{+1st}) \\ \vec{b}^T(y_i^{-1st}) \\ \vec{b}^T(y_i^{+2nd}) \\ \vec{b}^T(y_i^{-2nd}) \\ \vdots \\ \vec{b}^T(y_i^{+k_Tth}) \\ \vec{b}^T(y_i^{-k_Tth}) \end{bmatrix}\vec{\alpha}_i$$

$$i\vec{1} - \mathbf{H}_i\vec{p} \approx \mathbf{B}_i\vec{\alpha}_i \tag{11}$$

Equation (11) is the matrix form for the large matrix expression. Examination of the matrix $\mathbf{H}_i$ shows that each trajectory composes a 2×2 sub-matrix. Not

all output codes are excited by a particular trajectory due to the amplitude of the sinewave. If a trajectory does not excite state i then these equations are not included in (11).

Applying a least-squares solution [10] to (11) yields the following solution for the basis function coefficients.

$$\hat{\vec{\alpha}}_i \approx (\mathbf{B}_i^T \mathbf{B}_i)^{-1} \mathbf{B}_i^T (i\vec{1} - \mathbf{H}_i \vec{p}) \tag{12}$$

Equation (12) provides an expression for the desired basis function coefficients for the error at output code i. Clearly the coefficients are a function of the parameters, $\vec{p}$, since everything else is known. Keep in mind there are $2^n - 2$ of these equations but each equation has the same set of $\vec{p}$ parameters. The importance of (12) is that, as soon as the $\vec{p}$ are known, it is possible to determine basis coefficient estimates for the ADC error model.

3.2.2 Constrained Least-Squares Solution for the $\vec{p}$ Parameters

The final step in obtaining the estimated basis coefficients is to acquire an estimate of p_0 and p_1 for all trajectories. A proposed procedure is as follows.

Arbitrarily choose one trajectory, e.g. the jth, to use as a reference and use the independent-based method [7] (or any other method) to estimate p_0 and p_1. The independent-based method finds a solution for the estimate of p_0 and p_1 that minimizes the sum-squared expected error of (1). Equation (13) is obtained when the expected error terms are expanded over all output codes excited by the j^{th} trajectory.

$$\hat{\vec{E}} = \vec{v} - \mathbf{H}_{Ref} \cdot \begin{bmatrix} p_0^{jth} \\ p_1^{jth} \end{bmatrix} \tag{13}$$

where

$$\mathbf{H}_{Ref} = \begin{bmatrix} 1 & h_1^{+jth} \\ 1 & h_1^{-jth} \\ 1 & h_2^{+jth} \\ 1 & h_2^{-jth} \\ \vdots & \vdots \\ 1 & h_{2^n-2}^{+jth} \\ 1 & h_{2^n-2}^{-jth} \end{bmatrix}, \quad \text{and} \quad \vec{v} = \begin{bmatrix} 1 \\ 1 \\ 2 \\ 2 \\ \vdots \\ 2^n - 2 \\ 2^n - 2 \end{bmatrix} \tag{14}$$

A least-squares minimization of $\hat{\vec{E}}^T \hat{\vec{E}}$ yields the independent-based estimates, $\hat{p}_0^{jth}$ and $\hat{p}_1^{jth}$, for the reference trajectory.

$$\begin{bmatrix} \hat{p}_0^{jth} \\ \hat{p}_1^{jth} \end{bmatrix} = (\mathbf{H}_{Ref}^T \mathbf{H}_{Ref})^{-1} \mathbf{H}_{Ref}^T \vec{v} \tag{15}$$

The result in (15) provides an estimate of the unknown parameters for a single reference trajectory and this estimate is used as a constraint to find the remaining parameters.

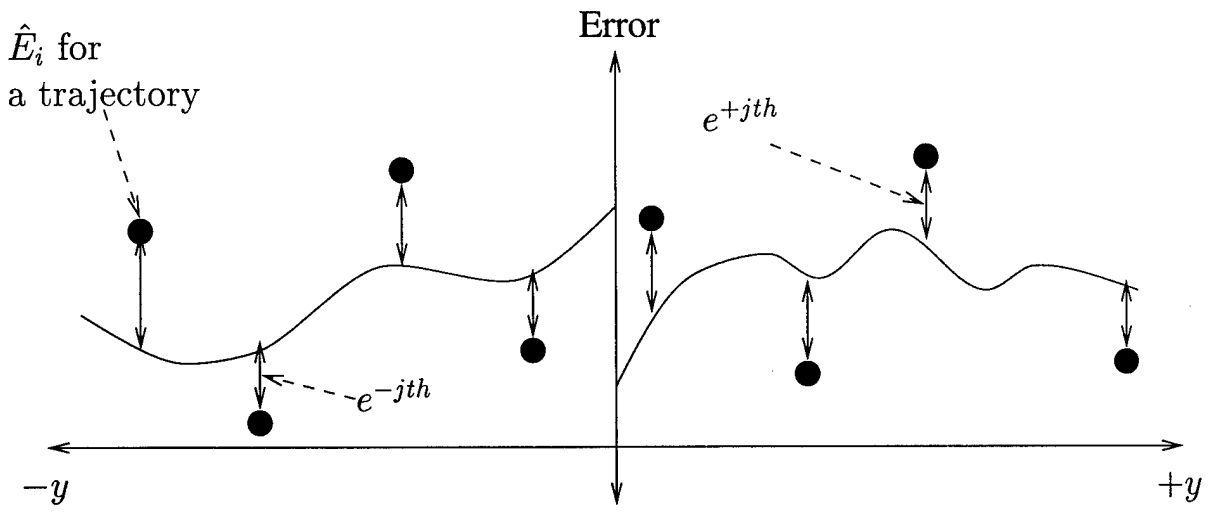

Figure 5: Theoretical error-vs-slope for output code i

It should be mentioned that other choices exist for finding estimates for the reference trajectory. One method is to just use ideal values for each parameter, e.g., $\hat{p}_0^{jth}$ is offset (usually equal to the number of states/2), and $\hat{p}_1^{jth}$ is the amplitude (in LSBs) of the test sinewave used for the jth trajectory. Another method is to use FFT analysis to determine A and c for the jth trajectory and then use the terminal based method to estimate G and x_0. A, c, G, and x_0 are sufficient to estimate the desired reference parameters through the definitions given in (8). Best success has been obtained using (15) when applied to a near full scale trajectory at a test frequency near the Nyquist limit for the calibration band of interest.

Fig. 5 illustrates an error versus slope behavior for a single output code i. Each trajectory yields two points on the plot, one on the left $(-y)$ and one on the right $(+y)$ side of the graph. This example shows a total of four trajectories intersecting at this state. The estimated basis coefficients determine the shape of the smooth curve which is supposed to provide minimum squared error in the residuals between the smooth curve and the data points *over all states of the ADC*. The distance from each point to the curve represents residual error from the error model for a given set of estimated basis coefficients. The points, except for the two points that correspond to the fixed jth reference trajectory, can be moved up or down vertically by varying $\vec{p}$. The estimate of $\vec{p}$ is selected to minimize the sum-squared residual error in the estimate of E_i (the distances between the points and curves of Fig. 5) over all output codes. The residual error estimate at the ith state, $\vec{\epsilon}_i$, is obtained by subtracting the model representation, $\mathbf{B}_i\vec{\alpha}_i$, from the estimated error (11) for all trajectories.

$$\begin{aligned} \vec{\epsilon}_i &= i - \mathbf{H}_i\vec{p} - \mathbf{B}_i\vec{\alpha}_i \\ &\equiv (\mathbf{I} - \mathbf{P}_{\mathbf{B}_i})(i\vec{1} - \mathbf{H}_i\vec{p}) \\ \text{where } \mathbf{P}_{\mathbf{B}_i} &= \mathbf{B}_i(\mathbf{B}_i^T\mathbf{B}_i)^{-1}\mathbf{B}_i^T \end{aligned} \tag{16}$$

$\mathbf{P}_{\mathbf{B}_i}$ is a projection matrix which has many useful properties, in particular, $\mathbf{P}_{\mathbf{B}_i}^n = \mathbf{P}_{\mathbf{B}_i}$. A constrained least-square estimate minimizes the sum-squared residuals, ϵ_i, under an applied constraint that p_0^{jth} and p_1^{jth}, of the reference trajectory, remain fixed. The constrained objective function is described by $\mathcal{L}(\vec{p})$ where the reference trajectory parameters are added as a constraint through the following vector definitions.

$$\begin{aligned} \mathcal{L}(\vec{p}) &= \sum_{i=1}^{2^n-2} \vec{\epsilon}_i^T\vec{\epsilon}_i + \vec{\lambda}^T[\vec{p} - \vec{p}_C] \\ \text{where } \vec{\lambda}^T &= \left[0\,0\cdots\lambda_1^{jth}\lambda_2^{jth}\cdots 0\,0\right] \\ \vec{p}_C^T &= \left[0\,0\cdots\hat{p}_0^{jth}\hat{p}_1^{jth}\cdots 0\,0\right] \\ \vec{p}^T &= \left[p_0^{1st}p_1^{1st}\cdots p_0^{jth}p_1^{jth}\cdots p_0^{k_T th}p_1^{k_T th}\right] \end{aligned} \tag{17}$$

Substitution of $\vec{\epsilon}_i$ into (17) and subsequent expansion yields (18) which shows explicit dependence of $\mathcal{L}$ upon the parameters, $\vec{p}$.

$$\begin{aligned} \mathcal{L}(\vec{p}) \equiv \sum_{i=1}^{2^n-2} i\vec{1}^T(\mathbf{I} - \mathbf{P}_{\mathbf{B}_i})i\vec{1} &- 2\sum_{i=1}^{2^n-2} i\vec{1}^T(\mathbf{I} - \mathbf{P}_{\mathbf{B}_i})\mathbf{H}_i\vec{p} \\ &+ \sum_{i=1}^{2^n-2} \vec{p}^T\mathbf{H}_i^T(\mathbf{I} - \mathbf{P}_{\mathbf{B}_i})\mathbf{H}_i\vec{p} + \vec{\lambda}^T[\vec{p} - \vec{p}_C] \end{aligned} \tag{18}$$

Now setting $d\mathcal{L}/d\vec{p} = 0$ and rearranging terms yields (19), a set of constraint equations for the parameters, $\vec{p}$.

$$\left[\begin{array}{c:c} \mathbf{C} & \mathbf{G} \end{array}\right] \left[\begin{array}{c} \vec{p} \\ \cdots \\ \frac{1}{2}\vec{\Lambda} \end{array}\right] = \vec{d} \tag{19}$$

$$\begin{aligned} \text{where } \mathbf{C} &= \sum_{i=1}^{2^n-2} \mathbf{H}_i^T(\mathbf{I} - \mathbf{P}_{\mathbf{B}_i})\mathbf{H}_i \\ \vec{d} &= \sum_{i=1}^{2^n-2} \mathbf{H}_i^T(\mathbf{I} - \mathbf{P}_{\mathbf{B}_i})\vec{1}i \\ \mathbf{G}^T &= \left[\begin{array}{cccccc} 0 & 0 & \ldots & 1 & 0 & \ldots \\ 0 & 0 & \ldots & 0 & 1 & \ldots \end{array}\right] \\ \vec{\Lambda}^T &= [\lambda_1\ \lambda_2] \end{aligned}$$

$\mathbf{C}$ is a square $2k_T \times 2k_T$ matrix and $\vec{d}$ is a column vector of length $2k_T$. The parameters $\vec{p}$ and Lagrange multipliers, $\vec{\Lambda}$, may be found by augmenting the constraint equations of (19) to obtain (20).

$$\begin{bmatrix} \mathbf{C} & \vdots & \mathbf{G} \\ \cdots & \cdots & \cdots \\ \mathbf{G}^T & \vdots & \mathbf{N} \end{bmatrix} \begin{bmatrix} \vec{p} \\ \cdots \\ \frac{1}{2}\vec{\Lambda} \end{bmatrix} = \begin{bmatrix} \vec{d} \\ \cdots \\ \vec{c}_{\hat{p}} \end{bmatrix} \tag{20}$$

Here, N is a 2×2 zero matrix and $\vec{c_{\hat{p}}}^T = \left[\hat{p}_0^{jth} \hat{p}_1^{jth}\right]$. Solving (20) yields estimates for the desired parameters, $\hat{\vec{p}}$, which can then be applied to (12) to obtain the error basis coefficients.

3.3 Calibration Algorithm Summary

The previous discussions have presented all mathematical details for developing the error model (7) of this paper. This section summarizes procedures for collecting calibration data and estimating basis function coefficients.

1. Drive the ADC with several sinewaves at different frequencies and amplitudes and collect sample sets for each signal. Perform the following steps for each sample set.
   (a) Obtain an associated slope estimate for the samples and sort into positive and negative slope sets.
   (b) Construct histograms from each sorted set and find h_i, from (8), for each output code observed for each histogram.
   (c) Find the average slope for each observed output code for each slope set.
   (d) Evaluate $\vec{b}^T(y)$, of (9), at each average slope, $y^{\pm kth}$, for both slope sets.
2. At this point, the ADC has been driven by all trajectories. The steps used to estimate the basis coefficients are as follows.
   (a) Find $\mathbf{C}$ and $\vec{d}$ of (20) by using (11) and (16) for each output code.
   (b) Select a jth reference trajectory, construct $\mathbf{H}_{Ref}$ of (13), and estimate the constrained parameters, $\hat{p}_0^{jth}$ and $\hat{p}_1^{jth}$, as in (15).
   (c) Obtain an estimate for the unknown $\hat{\vec{p}}$ parameters for all trajectories (based on the jth trajectory) through the inverse of (20).
   (d) Use $\hat{\vec{p}}$ to estimate the basis function coefficients as found in (12).

The basis function coefficients thus obtained can now be used to estimate ADC error at any state and input slope through the use of the dynamic error model (7).

4 Results

A method for obtaining a description of error as a function of ADC state and input slope for an ADC has been developed. Once the error model has been determined, estimated error can be created for any desired output code i and input slope y and used to compensate output samples to accomplish dynamic error correction. Compensating the output sample set of a calibrated ADC is straightforward. A sample set is obtained from the ADC along with corresponding slope estimates or measures. Take each element of the sample set, along with its associated slope value, and evaluate (7) to find the estimated expected error. Subtract this error from the current element and the result is a compensated sample.

The following sections present results of this dynamic error compensation procedure. Simulated and measured data are used to verify the effectiveness of the proposed error estimation and calibration procedure. All experimental and simulated results were obtained by calibrating the ADC with 50 trajectories containing 10 frequencies at 5 amplitudes per frequency. The frequencies uniformly spanned the first Nyquist band and the amplitudes were uniformly spaced from 50 to 95% of full scale loading. Estimated basis function coefficients, used to obtain the following results, were obtained by means of the method developed in Section 3.

Some comments are appropriate in regard to basis functions used to model error. There are no generating equations, or constraints, or other criteria that dictate the choice of the functions that should be used. Trial and error is one method and occasionally ADC architectural considerations provide useful criteria to assist in the choice of "natural" functions [4, 11]. For example: Folding architectures can cause periodic state dependent errors; many high-speed ADCs exhibit hysteresis; and wideband amplifiers used to drive comparator arrays introduce amplitude compression; all of which can be used to help guide the choice of basis functions to use for creating a meaningful error function. It should be noted that, when working with polynomials, it is best to keep the variable range between ± 1, hence, it is necessary to normalize slope values by an appropriate factor. Due to the usually different nature of negative slope errors compared to positive slope errors as pointed out in the discussion relating to Fig. 3, it is best to use two sets of functions to model slope dependent error for each state. One set is used for negative slope values and the other is used for positive slope values, thus allowing a discontinuous behavior across zero slope.

The results presented in this paper are all based on the use of 8 equally spaced Gaussian functions as described in Eq. (21). The spacing is determined by the maximum slope range and is the same for all states. The standard deviation, σ_k, was chosen to be equal to the separation in the means, y_k.

$$b_k(y) = exp(-((y - y_k)/\sigma_k)^2) \tag{21}$$

These functions are well-behaved for this algorithm and they are flexible in their inherent ability to represent a wide variety of functional behavior.

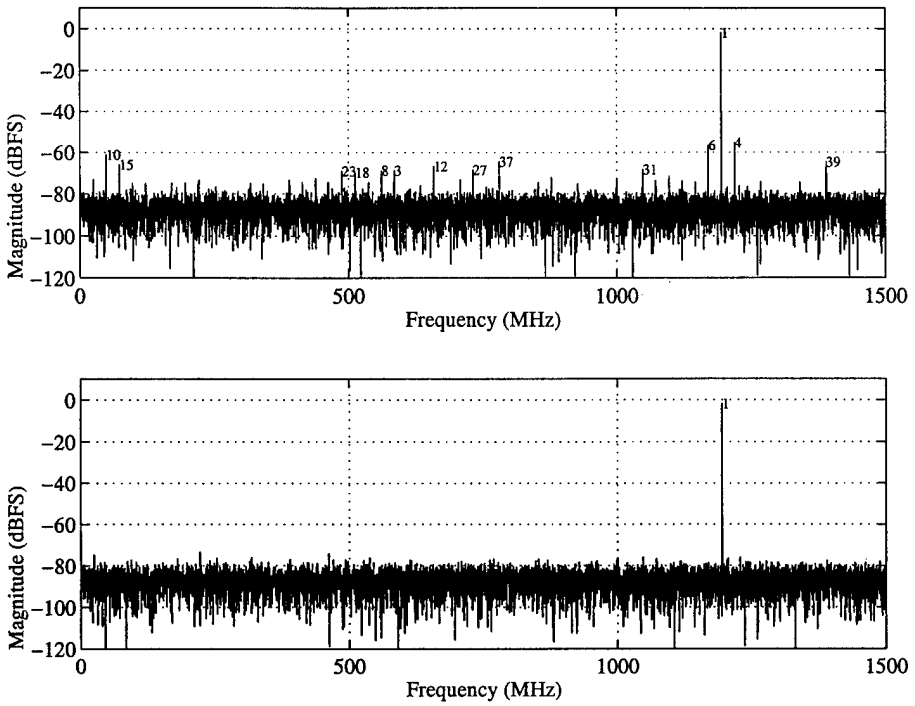

Figure 6: Uncompensated and compensated magnitude spectra, simulated ADC

4.1 ADC Simulation Results

The first result uses a simulated ADC to test the inherent ability of the algorithm to adequately estimate differential and high order errors. Figure 6 shows uncompensated and compensated magnitude spectra for a simulated 8-bit 3 GSPS folding ADC operated near Nyquist frequency with a nearly full scale test signal. The rather complex model contained only state and slope dependent errors in all mechanisms used to model the folding and interpolation operations of the ADC [11]. The estimated error function drives *all* harmonic distortion well into the noise floor. The graph illustrates the potential effectiveness of this error model and compensation technique. To date, no other dynamic compensation technique has provided the ability to remove such high order harmonic distortion from such a complex model. Tests at other frequencies and amplitudes have shown that the method provides equally excellent improvement over the entire Nyquist band.

The error associated with the uncompensated simulated result is strictly dependent upon output code and input signal slope. Simulated results demonstrate that this error model effectively removes all state and slope dependent error from uncompensated, but repeatable, ADC spectra.

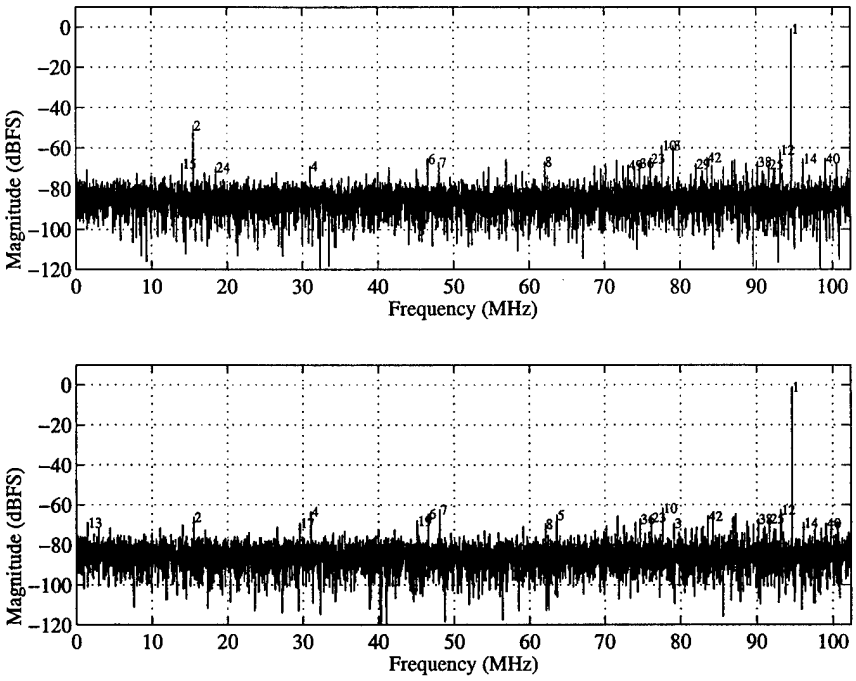

Figure 7: Uncompensated and compensated magnitude spectra, experimental results

4.2 Experimental ADC Results

Figure 7 shows uncompensated and compensated magnitude spectra for a real 8-bit flash ADC operated at 204.8 MSPS. The results are not as profound as the simulated results. Real ADCs contain errors that can not be accurately modeled as strictly a function of output code and signal slope but are functions of different parameters. Fig. 8 shows a spurious free dynamic range (SFDR) result for both compensated and uncompensated data for the ADC. (The SFDR measures the dB ratio between the fundamental signal magnitude and the magnitude of the largest harmonic, or spurious signal, over the full Nyquist band.) The curve shows excellent improvement in performance across the full Nyquist band. To date this algorithm has provided superior results compared to methods previously reported [8, 2].

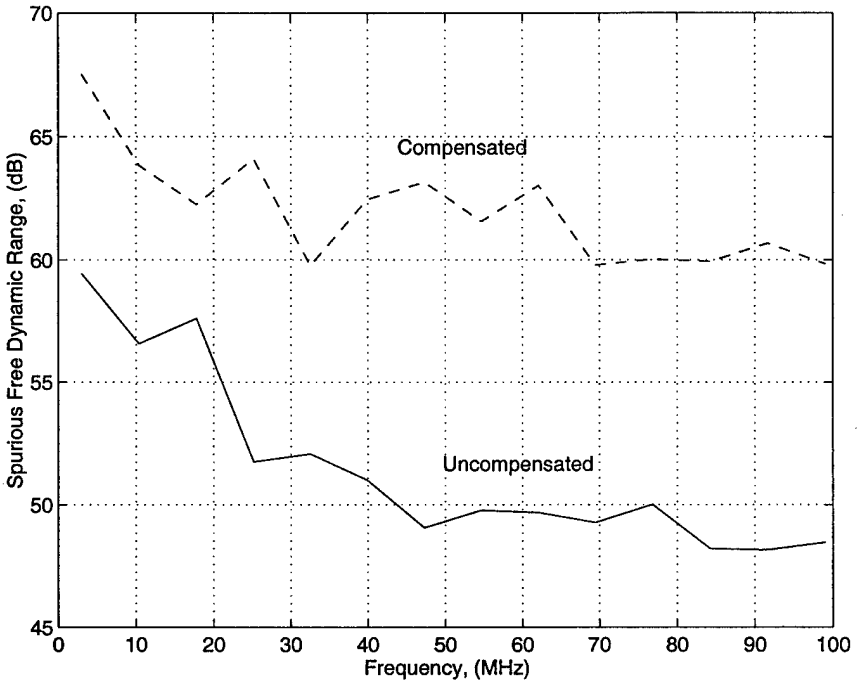

Figure 8: SFDR plot, experimental results

5 Discussion and Conclusions

This paper has presented complete details for implementing a state-slope dependent error model to determine dominant output code and slope dependent ADC error. A calibration method to obtain data that accurately describe an ADC according to the dynamic error model was developed. The validity of the error model was evaluated by means of error compensation on a simulated ADC whose error contained complex functions of state and slope only and experimental results were also obtained for a wideband ADC.

The high resolution of the error model accurately represents discontinuous expected error vs output code patterns as required by the example shown in Fig. (3) which illustrates typical behavior for wideband devices. Estimates of quantization thresholds were obtained using the sinewave histogram method and the histogram data were used in solving for the basis function coefficients at each output code. An important concept introduced in this paper is the use of the sorted sinewave histogram. The expected error differences obtained by this procedure dictate a necessity for defining slope dependent error models.

Once a dynamic error model was defined, a calibration method was developed to extract error data from the ADC when excited over its full range of output code and input slope. An array of sinusoidal signals with varying frequencies and amplitudes was passed through the ADC and output sample sets were processed. The amplitudes and frequencies were selected to fill the slope vs output code space of the ADC. The transformation of calibration data to an estimate of basis function coefficients was developed. By setting the dynamic error model for state i equal to the expected error of that state, a least-squares solution for the basis function coefficients as a function of the unknown parameters, $p_1 = GA$ and $p_0 = G(c - x_o)$, was determined.

The simulated 8-bit ADC test contained complex state and slope dependent error only. Results showed that this compensation method is highly effective for removing all errors that depend on state and slope. The results demonstrated by this method for real ADCs are not as profound as the results from simulating ideal performance, but important broadband improvements have been obtained which exceed previous compensation results for this type converter.

Previous experience has shown that it is difficult to model differential state-to-state errors to preserve the low frequency end of the SFDR while the high frequency end of the SFDR response is dependent upon hysteresis and the signal slope estimation procedure. The method presented in this paper handles both ends of the Nyquist band equally well due to the nature of the construction of the error model and its determining algorithm.

Exact mechanisms that contribute to ADC error are not always known and usually rely on more than just current state and slope. By studying residual errors which remain after this compensation, it will be possible, in future work, to study other mechanisms in more detail to determine residual error dependencies, e.g., effects from the previous state (track/hold) and bit dependent feedback anomalies from pipelined digital outputs on the ADC substrate (digital "kickback").

References

[1] J. Larrabee, D. Hummels, and F. H. Irons, "Adc compensation using sinewave histogram methods," in *Proceedings of IEEE International Instumentation and Measurement Technology Conference*, (Ottawa), May 1997.

[2] D. Hummels, F. Irons, R. Cook, and I. Papantonopoulos, "Characterizaion of ADCs using a non-iterative procedure," in *Proceedings of IEEE International Symp. on Circuits and Systems*, (London), May 1994.

[3] I. Papantonopoulos, "Error modeling for folding and interpolating analog to digital converters," Master's thesis, University of Maine, Dept. of Electrical and Computer Eng., Orono Maine, Aug. 1995.

[4] F. Irons, D. Hummels, and I. Papantonopoulos, "ADC error diagnosis," in *Proceedings of IEEE International Instumentation and Measurement Technology Conference*, (Brussels), June 1996.

[5] W. Ahmed, "Fast orthogonal search for training radial basis function neural networks," Master's thesis, University of Maine, Dept. of Electrical and Computer Eng., Orono Maine, Aug. 1994.

[6] R. Gray and T. Stockham, Jr., "Dithered quantizers," *IEEE Trans. Inform. Theory*, vol. IT-39, pp. 805–812, May 1993.

[7] "IEEE Standard 1057, standards for digitizing waveform recorders," tech. rep., National Institute of Standards and Technology, July 1989.

[8] F. Irons, D. Hummels, and S. Kennedy, "Improved error compensation for analog-to-digital converters," *IEEE Trans. Circuits and Systems*, vol. 38, pp. 958–961, June 1991.

[9] D. Hummels, I. Papanonopoulos, and F. Irons, "Identification of error mechanisms in a folding and interpolating ADC," in *Proceedings of IEEE International Symp. on Circuits and Systems*, (Atlanta), May 1996.

[10] L. Scharf, *Statistical Signal Processing: Detection, Estimation, and Time Series Analysis.* Reading, Massachusetts: Addison-Wesley, 1991.

[11] F. Irons, D. Hummels, and C. Zoldi, "ADC architectural diagnostic testing procedures," in *Proceedings of Government MicroCircuit Applications Conference*, (Orlando), Mar. 1996.

A Virtual Instrument Bus Using Network Programming

D.J. Rawnsley, D.M. Hummels, B.E. Segee
University of Maine
Orono, Maine *

Abstract— *This paper provides an overview of a virtual instrument bus created at the University of Maine Orono. Software to support automated tests has become difficult to maintain as the number of test boards and test instruments grows. A variety of test instruments such as logic analyzers, signal generators, and data caches connect and communicate to workstations using a General Purpose Interface Bus (GPIB).*

This paper describes two software packages. The first is a "virtual instrument bus" that makes a large number of GPIB buses on separate networked computers appear to be on a single bus. The second is an object-oriented instrument library. The Library is designed to support a variety of instruments using a common framework in an easily maintained software package.

The virtual instrument library is developed using remote procedure calls (RPC). All workstations supporting an instrument bus run a background program called a Bus Server that handles bus communications and provides an interface to the computer network. The Bus Server can be programmed to handle any kind of bus, not just the GPIB.

Communication to the various Bus Servers is handled by the Virtual Bus Library. This interface makes the physical configuration of the instrument buses transparent to the software developer. The library supports a small set of routines modeled after the IEEE 488.2. It also provides searching functions for locating specific instruments on the computer network, and maintains a list of all machines that have instrument buses connected to them.

The virtual bus software provides easy code reuse for quick program generation used for automated testing, at the same time making all instruments appear to be located on one single bus. This software will greatly facilitate the future development of complex experiments requiring multiple bus instrument coordination.

This work has been supported in part by the ARPA HBT/ADC program under a contract administered by the Office of Naval Research Grant N000149311007 and the DEPSCoR program through the Army Research Office Grant DAAH04-94-G-0387

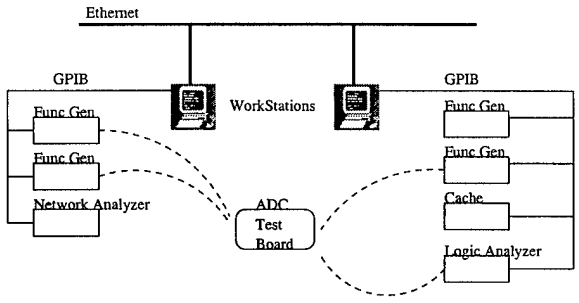

Figure 1: Ideal Lab Configuration

I. Introduction

This paper presents the development and implementation of instrument control software for use in a networked computer environment. The project was motivated by ongoing research in the Communication Laboratory at the University of Maine. The Communications Lab, among other things, analyzes Analog to Digital (A/D) converter output to provide a means of compensation for the error introduced by the device. Software to support automated tests for data acquisition from A/D test boards has become difficult to maintain as the number of test boards and test instruments grows. A variety of test instruments such as logic analyzers, signal generators, and data caches connect and communicate to workstations using a General Purpose Interface Bus (GPIB). Software to control test instruments that are physically located on separate workstations within the lab as illustrated in Figure 1 are extremely time consuming or impossible to configure. Moving instruments from one workstation to another required reconfiguring software and recompiling an extensive software package.

The software maintenance and network support issues encountered on the Communications Lab are typical of those encountered when instruments are controlled over a computer network. While users have become accustomed to distributed network resources (shared file systems, transparent access to printers, etc) instrument con-

trol software has not supported the capabilities of most networks. For a networked computing environment, instrument control software should support the following features:

- Instruments should be portable to any machine on the local network without recompiling test software.
- Test software should not be platform dependent. Tests should operate correctly regardless of the platform that the test is run from.
- Development of test software should not be platform dependent. Once the network instrument control libraries are compiled for a particular architecture, the test software should be supported for any machine using that architecture.
- The software interface should be consistent regardless of the physical instrument bus interface.
- A common software interface should be provided for instruments with common functionality. For example, all function generators should respond to a common set of amplitude/frequency configuration commands.

II. Existing Software

Existing software used in the Lab for data acquisition and controlling instruments is written in the C programming language. The physical addresses of the test instruments and the names of the machine hosts that they are connected to are hard-coded into the software. In order to move instruments from one machine to another, or to change its address, the existing software package has to be recompiled for the changes to take effect.

To access instrument software for a specific instrument, the user has to be logged-on to the workstation to which the instrument is physically connected. Automated tests involving multiple instruments connected to different physical buses cannot be supported. In order to incorporate an instrument on a different bus than the one the test is running on, the cabling would have to be physically changed to the new bus. The address of the instrument would have to be set so that it did not conflict with any other instrument on the new bus location, and the software would have to be recompiled to reflect these changes. Setting up for such changes is time consuming and problematic.

With the expansion of our facility to include new high speed instruments for A/D testing, the current setup is not an efficient use of equipment.

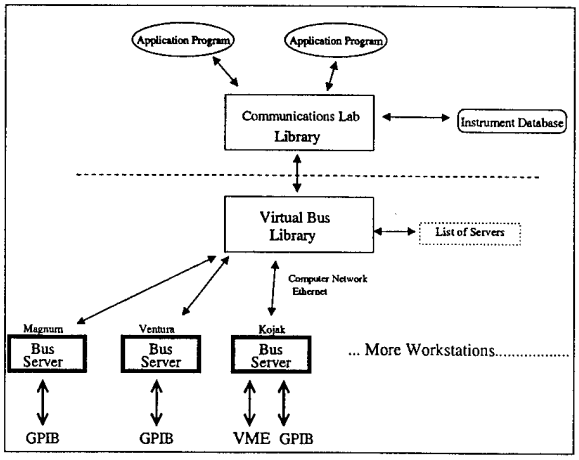

Figure 2: Virtual Bus Block Diagram.

III. The Approach

Two software packages are described which together address these issues. The first is a "virtual instrument bus" which makes a large number of physical buses on a computer network look like a single bus. The Virtual Instrument Library is designed to support the computer network communications making the computer networking transparent to program developers.

The second software package is an object-oriented instrument library which is specific to instruments within the communications lab. The Communications Lab Library is designed to support a variety of instruments using a common framework in an easily maintained software package. The communication between the libraries is illustrated in Figure 2.

A. Virtual Instrument Library

The virtual instrument library is developed using remote procedure calls (RPC). RPC is a mechanism for building a distributed system of programs that handle all communications between the physical buses, the workstations, and the network. All workstations supporting an instrument bus run a background program that handles all bus communications (like GPIB) and provides an interface to the computer network. These programs are shown in Figure 2 as Bus Servers. The Bus Server can be programmed to communicate with instruments using any kind of bus (not just a GPIB).

Communication to the various Bus Servers is handled by the Virtual Bus Library. This interface makes the physical configuration of the instrument buses transparent to the data acquisition software developer. The library supports a small set of routines modeled after the IEEE 488.2 GPIB standard. It also provides searching functions for lo-

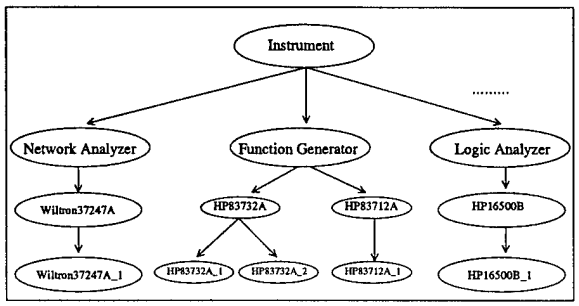

Figure 3: Communications Lab Library Software Structure.

cating specific instruments on the computer network, and maintains a list of all machines that have instrument buses connected to them. This library handles all communications to the RPC Bus Servers, and is the interface to the computer network for the Communications Lab library.

B. Communications Lab Library

The Communications Lab library is created using an object-oriented architecture design. It is designed to represent the functionality of the test instruments and provide simplified software reuse and changeability in a modularized fashion. The structure of the library is illustrated in Figure 3.

Object-oriented programming is a method of extending abstract data types to allow for type/subtype relationships among data types. In C++ this is accomplished with inheritance. Instead of re-implementing shared characteristics, an object can inherit the functionality of the class it was derived from. The C++ class mechanism allows programmers to define their own data type.

The Communications Lab library uses C++ inheritance extensively. Each level in Figure 3 inherits the functionality of the level above it. All test equipment are bus instruments that have common `Instrument` functions. A piece of test equipment, such as a `Function Generator`, has its own common functions to complement the common `Instrument` functions. For example, every `Function Generator` supports a common software interface for controlling the frequency or amplitude of the generator. A specific generator is a Hewlett Packard 83732a which has a variety of functions that are provided which are specific to that model.

The Communications Lab library maintains a bus configuration database shown in Figure 2 which is automatically updated if a change to an instrument's address or location is detected. When one of these instruments, like an `HP83732a`, is used in a program, the library first checks the current location and address in the instrument database. This is done to make sure the program is talking to the correct instrument. If not, search functions of the Virtual Bus library are run to locate the test instrument and update the instrument database of the new test instrument location and address.

The virtual bus software provides easy code reuse for quick program generation used for automated testing, at the same time making all instruments appear to be located on one single bus. This software will greatly facilitate the future development of complex experiments requiring multiple bus instrument coordination.

IV. Virtual Bus Software Architecture

This section gives a quick overview of the software architecture including the names and purposes of the major executables and routines. Figure 4 shows the client-server architecture used for the Virtual Bus Software.

A. Client Side

The Application Programs are the client side of the architecture. All Application programs use the two software libraries, the Communications Lab Library and the Virtual Bus Library, to create client executables. The Communications Lab Library is an object-oriented library that models types of instruments, and communicates with the Instrument Database Server for up to date information on instrument locations. The Virtual Bus Library is the interface to the network communications. This interface is used by the Communications Lab Library to provide reusable objects for Application Programs.

A.1. Virtual Bus Interface

The interface for the virtual bus abstracts away the ideas of network programming from the Communications Lab Library and Application Programs. All interface functions establish connections with the specified servers and handle network communications. When completed, each routine disconnects from the server. Each routine provides an interface that makes it appear that the routine is running locally. When in fact, it maybe executing on a different workstation. The following is brief review of each interface routine.

1. **v_send()**: Send commands or data to a specified instrument.

2. **v_receive()**: Receive data from a specified instrument.

3. **v_bustimeout()**: Set the timeout value for the physical bus. The timeout value is the approximate minimum length of time that I/O functions can take before a timeout occurs.

4. **v_findlisteners()**: Poll the bus to find the number of listeners.

There are two helper functions that are used by `v_findlisteners()`:

1. **get_valid_addresses()**: Build a list of addresses for the `v_findlisteners()` function.
2. **gethosts()**: Get a list of host workstations and possible bus addresses from a configuration file.

B. Server Side

Two different types of servers are used for the virtual bus: the Instrument Server and the Instrument Database Server.

B.1. Instrument Server

The Instrument Server, also called the Bus Server, is run as a background process which is configured by the `startgpibd` executable. When this process is started during workstation boot-up, it is replaced with the `gpibd` executable. `gpibd` is the server that handles all client requests to communicate with the instrument bus. When a connection is made, a specific service is performed by calling one of the following routines:

1. **v_send_1()**: Send commands or data to a specified instrument physically connected to the same workstation this procedure is executed on.
2. **v_receive_1()**: Receive data from a specified instrument physically connected to the same workstation this procedure is executed on.
3. **v_bustimeout_1()**: Sets the timeout value for the local bus.
4. **v_findlisteners_1()**: Poll the local bus to find the number of listeners.

Each one of these routines calls vender specific GPIB interface software to communicate on the bus.

B.2. Instrument Database Server

The Instrument Database Server is run as a background process which is configured by the `startcommd` executable. When this process is started during workstation boot-up, it is replaced with the `commd` executable. The `commd` server handles all client requests for information about the location of a specific instrument. This server provides two database services:

1. **locate_1()**: Given an instrument identifier, return the last known location of that instrument.
2. **update_1()**: Update the location of an instrument in the database to the current location.

There maybe as many Instrument Servers as there are workstations that have external buses, but only one Instrument Database Server is needed to maintain instrument locations.

V. Conclusions

The Virtual Instrument Bus software has proven to be an excellent software package for data acquisition across a local network. The convenience of running and creating data acquisition software from any workstation on the network makes development easy for the user. The ease of moving instrument locations and changing instrument addresses for specific test setups without recompiling software allows for easy configuration of automated tests. Once an instrument has had its location or address changed the software will update the database so that no searching will take place the next time the software is run. The Virtual Instrument Bus software is a powerful tool for providing development of complex experiments requiring multiple bus instrument coordination.

Biographies of the Authors

Daryl Rawnsley received a B.S. in Computer Engineering at the University of Maine in 1995. He is a candidate for the Master of Science degree in Computer Engineering from the University of Maine in May 1997. His current research interests include communications and computer networks. He is a member of IEEE, Eta Kappa Nu, and his interests include firefighting and sports.

Don Hummels is an Associate Professor of Electrical and Computer Engineering at the University of Maine. He has been with the University of Maine since 1988. He obtained his B.S. degree from Kansas State University, and his M.S. and Ph.D. degrees from Purdue University, all in Electrical Engineering. His research interests include nonlinear signal processing, and characterization and compensation of nonlinear components used in communications receivers.

Dr. Bruce Segee received a PhD in Engineering from the University of New Hampshire in 1992. He has been an assistant professor of electrical and computer engineering at the University of Maine since that time. At the University of Maine he heads the Instrumentation Research Laboratory, an organization dedicated to research and teaching involving instrumentation and automation. Work in the lab includes the use of PC's, PLC's, and embedded controllers for instrumentation, automation, and networking. Work also includes the use of fuzzy logic and artificial neural networks.

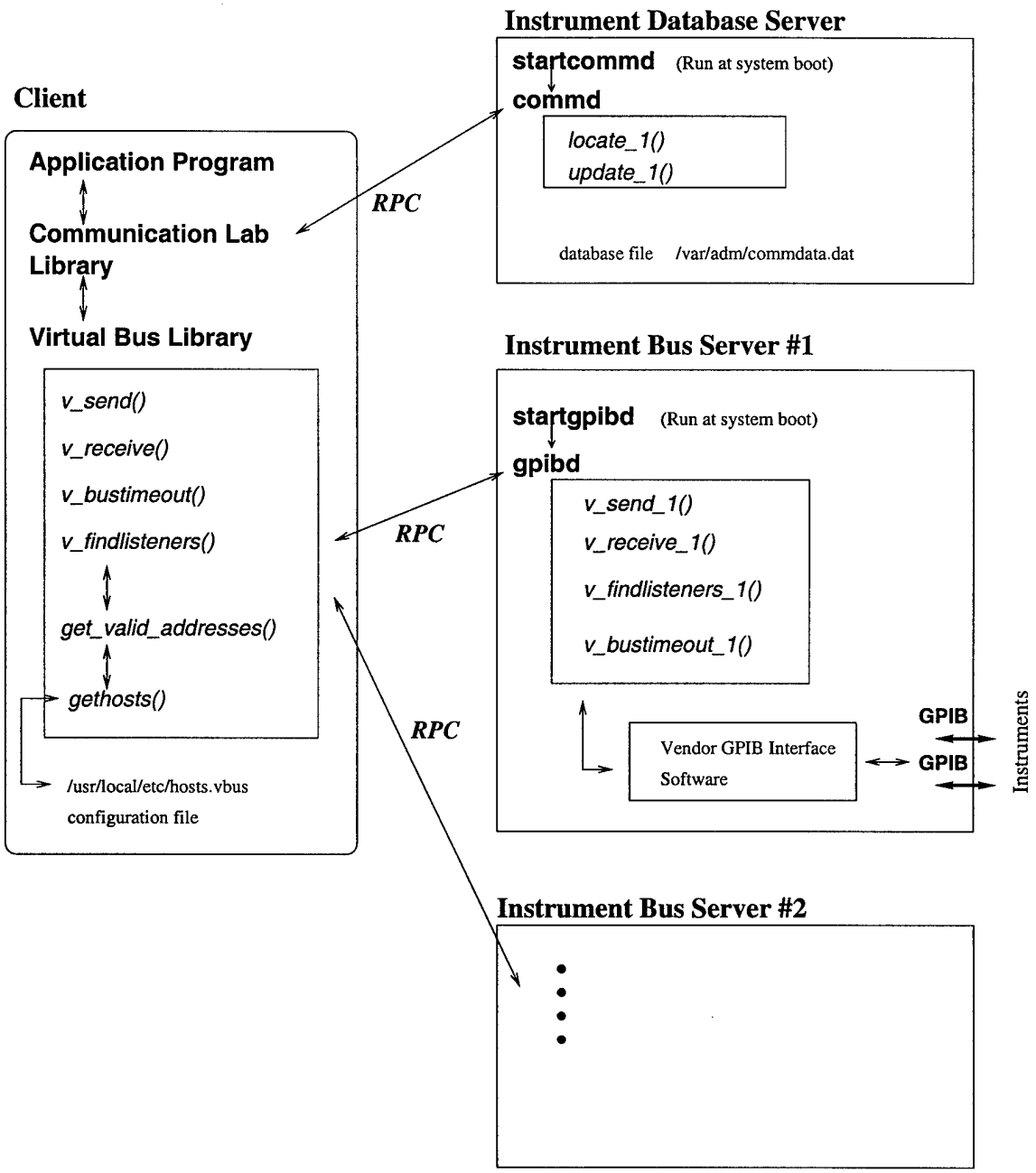

Figure 4: Software Architecture

IEEE Instrumentation and Measurement
Technology Conference
Ottawa, Canada, May 19-21, 1997

A Compensation Technique for Sigma-Delta Analog-To-Digital Converters

D.M. Hummels, D. Gerow, F.H. Irons
Electrical and Computer Eng, University of Maine
Orono, Maine 04469
(207) 581-2245 *

Abstract— *This paper explores a compensation technique for mismatched amplifier gain and capacitor values in a three stage, third order noise shaping Sigma-Delta analog-to-digital converter (ADC). The paper concentrates on multistage noise shaping (MASH) architectures. Two possible sources of distortion are examined and simulated. Using these simulation results, the distortion is identified, and a modified architecture is designed that attempts to compensate for these distortion terms. Simulation results show that the distortion caused by finite Op-amp amplifier gains (all near 60 dB) may be reduced by over 13 dB over the operating band of the converter by using the modified architecture.*

I. Introduction

The recent need for high resolution Analog-to-Digital Converters (ADCs) has forced the communications industry to research new radical designs to overcome the physical limitations of classic flash and other Nyquist rate converters. One of the new designs emerging as a candidate for low frequency quantization, such as in digital audio or Integrated Services Digital Network (ISDN) applications [1], is the Sigma-Delta converter. The Sigma-Delta converter utilizes oversampling and feedback loops to produce high resolution output samples using multiple quantizers, each with a low number of bits. This allows relaxed tolerances for each individual quantizer, while still producing the desired high resolution output samples.

This paper examines the third order noise shaping Sigma-Delta modulator, and a simulation of this modulator is given. Portions of this simulation are described in detail in section II. where two specific error mechanisms commonly found in Sigma-Delta modulators are examined. Simulation results lead to a method in which the architecture of the modulator may be modified to compensate for distortion produced by these error mechanisms. A least-squares calibration procedure is then used to calibrate the required filter coefficients for the simulated converter. These procedures lead to significant reductions in the distortion levels for sigma-delta converters which are limited by mismatches in finite op-amp gains or capacitor values in switched capacitor implementations.

This work has been supported in part by the ARPA HBT/ADC program under a contract administered by the Office of Naval Research Grant N000149311007 and the DEPSCoR program through the Army Research Office Grant DAAH04-94-G-0387

II. Error Modeling

This paper concentrates on multistage noise shaping (MASH) architectures [1, 2]. Figure 1 shows a block diagram of a 3-stage modulator. The modulator is comprised of three first order Sigma-Delta loops assembled in a feedforward architecture. In this figure, the m-bit ADCs are replaced by additive white noise source models. This substitution is accurate if the quantization noise from the ADC is independent from the input signal and uncorrelated from sample to sample [3]. Although this statement is not accurate for quantizers with low numbers of bits, the model does lead to useful design guidelines for Sigma-Delta converters [3]. Exact analysis for low resolution quantizers is difficult, and usually requires simulation.

By summing the three modulator outputs after they are passed through their respective loop filters, the final modulator output can be written as

$$\begin{aligned} Y_M(z) &= z^{-3}X(z)\ldots \\ &\quad +z^{-2}(1-z^{-1})N_{ADC1}(z)\ldots \\ &\quad -z^{-2}(1-z^{-1})N_{ADC1}(z)\ldots \\ &\quad +z^{-1}(1-z^{-1})^2N_{ADC2}(z)\ldots \\ &\quad -z^{-1}(1-z^{-1})^2N_{ADC2}(z)\ldots \\ &\quad +(1-z^{-1})^3N_{ADC3}(z)\ldots \end{aligned} \quad (1)$$

$$= z^{-3}X(z)+(1-z^{-1})^3N_{ADC3}(z). \quad (2)$$

The $(1-z^{-1})^3$ term is a high-pass function, attenuating the low frequency portion of the quantization noise. Since the converter is also oversampled, the input signal is band-limited to this low frequency range, and after passing this output through several low-pass filters and decimators, the result is a high resolution digital representation of the low

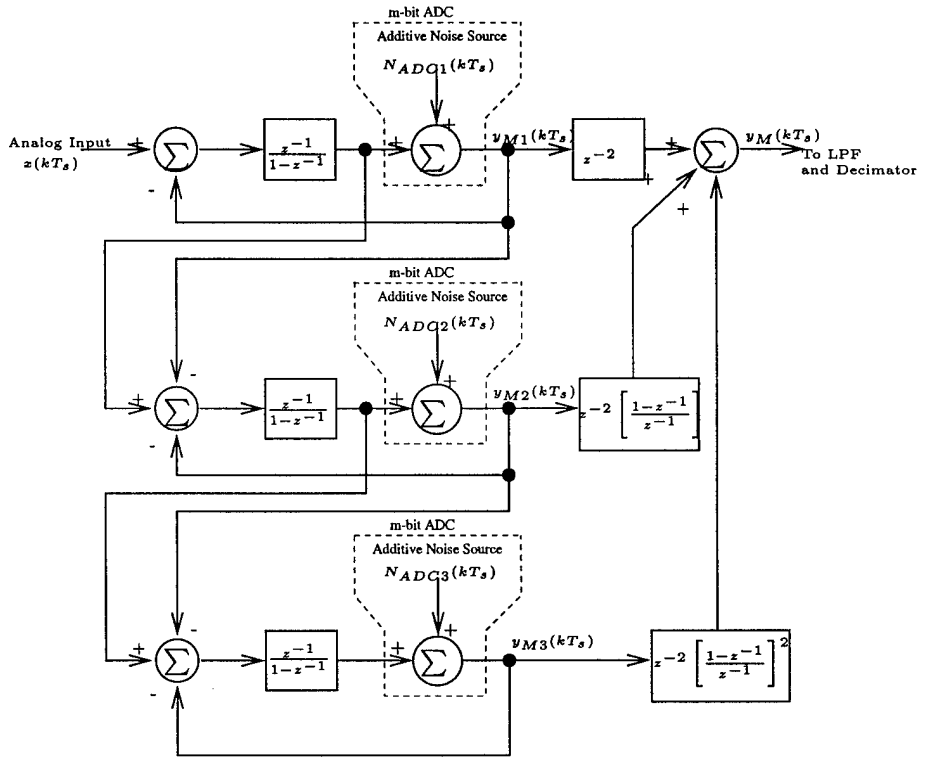

Figure 1: Third Order Multistage Sigma-Delta Block Diagram

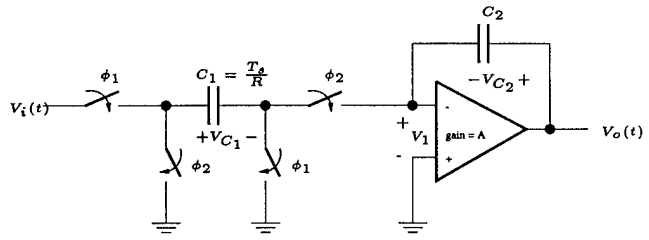

Figure 2: Non-Inverting Switched Capacitor Discrete-Time Integrator

frequency analog input signal. Note that the quantization noise in the output samples is a function of the quantization noise introduced in the third loop only, assuming that the cancellation of terms seen in (2) is exact. For non-ideal loop filters or mismatched first order loops, this cancellation will not be exact, resulting in distortion in the final modulator output.

The effects of non-equal finite amplifier gains or capacitor mismatches may be modeled by examining a switched capacitor implementation of the discrete-time integrators (shown as $z^{-1}/(1-z^{-1})$ blocks in Figure 1). Figure 2 shows a typical implementation. The discrete-time model for the switched capacitor integrator may be shown to have transfer function

$$\frac{V_o(z)}{V_i(z)} = \frac{\left[\frac{AC_1}{AC_2+C_2+C_1}\right] z^{-1}}{1-\left[\frac{AC_2+C_2}{AC_2+C_2+C_1}\right] z^{-1}}. \qquad (3)$$

This equation allows a simulation to model switched capacitor discrete-time integrators, such as those found in Sigma-Delta loops, given specific component values and amplifier parameters. It should be noted that when $C_1 = C_2$, and as the open loop amplifier gain approaches infinity, this model approaches the ideal discrete-time model for an integrator.

Simulating these parameters is accomplished by substituting finite amplifier open loop gain values or finite capacitor values, or both, into the switched capacitor integrator equation given in (3). Typical values for these parameters are 10 pF for the capacitor values and 60 to 80 dB for the open loop amplifier gains [1, 2, 4]. By examining (3), one can see that the same results are obtained by either equating the capacitor values and assigning a finite amplifier gain, or by allowing the amplifier gain to approach infinity and assigning unequal capacitor values: a non-unity scalar coefficient is present on each delay term. Although this error mechanism does not result in a nonlinear func-

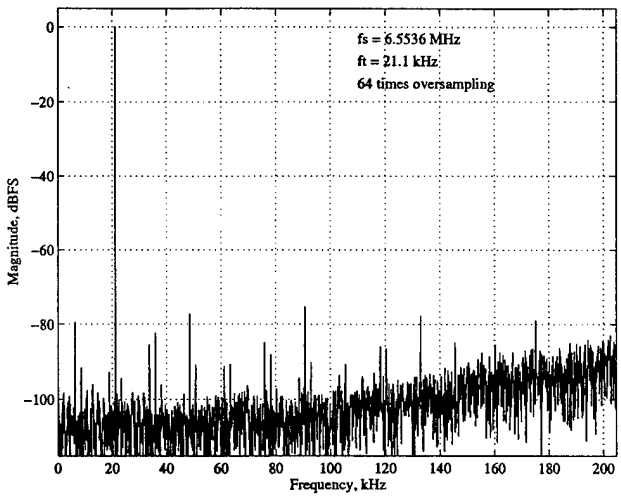

Figure 3: Sample spectrum of a simulated 3-stage MASH converter with finite amplifier gains.

tion, it does result in nonlinear distortion in the output spectrum for each Sigma-Delta loop due to the nonlinearity introduced by the quantizer when the quantizer is not replaced by a uniform white noise source. The distortion in each loop is removed from the output spectrum of multistage converters to some extent through the cancellation of quantization noise, as seen in (1). However, some distortion remains, as seen in Fig. 3, where a sinusoid with no harmonics is passed through the three stage, third order noise shaping Sigma-Delta modulator simulation, with the three amplifier open loop gains set to unequal, finite values ($A_1 = 1000$ V/V, $A_2 = 1020$ V/V, and $A_3 = 1015$ V/V), and with all equal capacitor values. This distortion is also present when the amplifier gains are equal and very small, which is attractive due to the reduction in power consumption for a lower gain device. Use of low-gain amplifiers could be made more practical by developing procedures to remove the resulting distortion. The process of removing this distortion is referred to as compensation.

III. Error Compensation

Errors from capacitor mismatches or finite amplifier gains has been shown to result in degradation in the performance of the converter. By adjusting gains in the recombination of the modulator stages, performance can be significantly improved. One architecture that uses programmable filters is shown in Figure 4. Test points are included in the modified architecture to allow the device to be calibrated. In Figure 4, the nominal transfer functions of the recombination filters from the first two sigma-delta modulators of Figure 1 have been replaced with programmable filters. Let the transfer functions of these filters be denoted $G_1(z)$ and $G_2(z)$.

To solve for the filter coefficients using a least squares curve fit approach, the ideal loop filter equation must be rewritten in terms of the nominal loop filter coefficients:

$$z^2 G_1(z) = (1 + \epsilon_1) - (\epsilon_2) z^{-1} \tag{4}$$

and

$$z G_2(z) \doteq (1 + \epsilon_3) - (1 + \epsilon_4) z^{-1} - (\epsilon_5) z^{-2}. \tag{5}$$

The third loop filter, $G_3(z)$, need not be rewritten since the third loop filter coefficients do not contribute to output distortion and therefore are not calculated. The Sigma-Delta modulator output equation is now written

$$\begin{aligned} Y_M(z) \;=\; & z^{-2}\left[(1+\epsilon_1)Y_{M1}(z) - (\epsilon_2)z^{-1}Y_{M1}(z)\right] \ldots \\ & + z^{-1}\left[(1+\epsilon_3)Y_{M2}(z) - (1+\epsilon_4)z^{-1}Y_{M1}(z) \ldots\right. \\ & \left. - (\epsilon_5)z^{-2}Y_{M2}(z)\right], \end{aligned} \tag{6}$$

Using data collected from the first two loop outputs, $y_{M1}(kT_s)$ and $y_{M2}(kT_s)$, and creating the output sequence as defined in (6) for the case $\epsilon_i = 0$, $i = 1, 2, \ldots, 5$, an output spectrum may observed that contains all the distortion terms that are seen in the final converter output spectrum when the nominal loop filter coefficients are used. The distortion terms may be identified as intermodulation distortion between the input signal frequency and one-half the sampling frequency. To identify effective filter coefficients, the power in the distortion terms is minimized by adjusting $\vec{\epsilon} = [\epsilon_1\ \epsilon_2\ \epsilon_3\ \epsilon_4\ \epsilon_5]^T$. A least squares method of curve fit is used to find the best set of coefficients that minimize the total distortion power.

IV. Results

Figures 3 and 5 illustrate the potential performance of the error compensating architecture. The figures show output spectra for the two architectures for a pure sinusoidal input signal. In this case, the distortion introduced by the converter is caused by finite Op-amp gains (all near 60 dB). Use of compensating architecture resulted in a reduction in distortion by over 13 dB within the $0 < f < 51.2$ kHz output band for this converter.

V. Summary

This paper has presented a method in which a three stage, third order noise shaping Sigma-Delta modulator architecture can be modified to remove output distortion due to mismatched capacitor or amplifier open loop gain values. The modulator structure was first introduced, and a simulation based on this architecture was given. Two dominant error mechanisms in the switched capacitor integrators of the modulator were explored, and were added to the simulation module. After simulation results showed that these

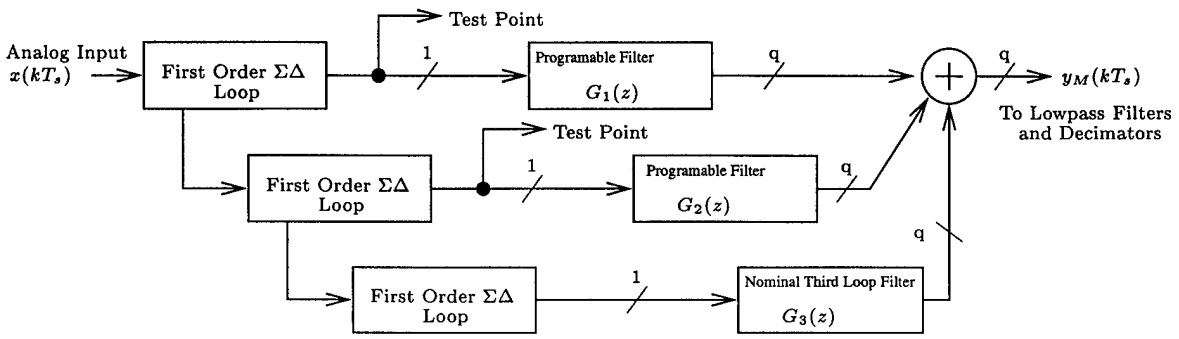

Figure 4: Compensating Three Stage Sigma-Delta Architecture

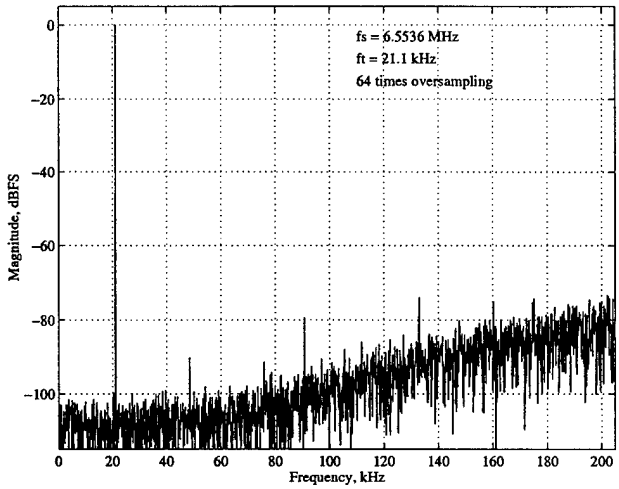

Figure 5: Sample spectrum of a simulated 3-stage compensating MASH converter with using estimated coefficients to compensate for finite amplifier gains.

error mechanisms produced output distortion, the architecture was again analyzed, resulting in modified loop filters that matched the nonlinearities of the integrators. Loop filter coefficients were calculated without prior knowledge of the error parameters, since such parameters are not known after fabrication of the device. Simulation results have illustrated the potential benefits of the procedure.

References

[1] B.P. Brandt and B.A. Wooley, "A 50-MHz Multibit Sigma-Delta Modulator for 12-b 2-MHz A/D Conversion," *IEEE Journal of Solid-State Circuits*, vol. 26, no. 12, pp. 1746-1755, December 1991.

[2] Y. Matsuya, K. Uchimura, A. Iwata, T. Kobayashi, M. Ishikawa, and T. Yoshitome, "A 16-bit Oversampling A-to-D Conversion Technology Using Triple- Integration Noise Shaping," *IEEE Journal of Solid-State Circuits*, vol. 22, no. 6, pp. 237-244, December, 1987.

[3] J.C. Candy and G.C. Temes, *Oversampling Methods for A/D and D/A Conversion*, New York: IEEE Press, 1992.

[4] S.R. Norsworthy, I.G. Post, and H.S. Fetterman, "A 14-bit 80-kHz Sigma-Delta A/D Converter: Modeling, Design, and Performance Evaluation," *IEEE Journal of Solid-State Circuits*, vol. 24, no. 2, pp. 256-265, April, 1989.

IEEE Instrumentation and Measurement
Technology Conference
Ottawa, Canada, May 19-21, 1997

ADC Compensation Using A Sinewave Histogram Method

J.H. Larrabee, D.M. Hummels, F.H. Irons
Electrical and Computer Engineering, University of Maine
Orono, Maine 04469
(207) 581-2245 *

Abstract— *The need for high dynamic range analog to digital converters (ADCs) is strong due to the growing use of digital signal processing in communications applications. ADC performance has improved over the years due to advances in both design concepts and semiconductor technologies. However most ADCs still still exhibit undesirable distortion in addition to quantization error, thus limiting spurious free dynamic range, (SFDR), in digital receiver applications. This paper describes a new method for developing ADC error function models using standard sinewave histogram methods. The method provides better error representation by providing error basis functions for every state. Results show this method is capable of removing all slope dependent errors in complex ADC error models and application to a 200 MSPS ADC showed more than 10 dB improvement in SFDR over the full nyquist band for the ADC.*

I. Introduction

This paper describes a new compensation method for finding ADC error functions. The method is based upon expected error for each state of the converter and the error for each state is estimated as a function of the slope of the input to the ADC. Sinewave histogram techniques are used to obtain estimates of expected error and these estimates are used as a basis to curve fit error-versus-slope for each state. This paper also describes a calibration technique that obtains an accurate model for the ADC. Some simulated and experimental results using this compensation scheme are described.

II. ADC Error Modeling

ADC performance can be described by the transfer function of output code versus input voltage [1]. Expected error is a common ADC error parameter and for a given state, (ith output code), it is defined as the average error at state i.

* This work has been supported in part by the ARPA HBT/ADC program under a contract administered by the Office of Naval Research Grant N000149311007 and the DEPSCoR program through the Army Research Office Grant DAAH04-94-G-0387

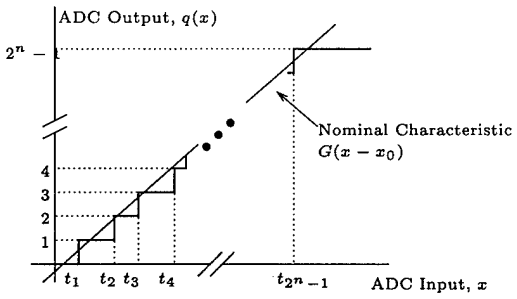

Figure 1: Non-Ideal ADC Transfer Function

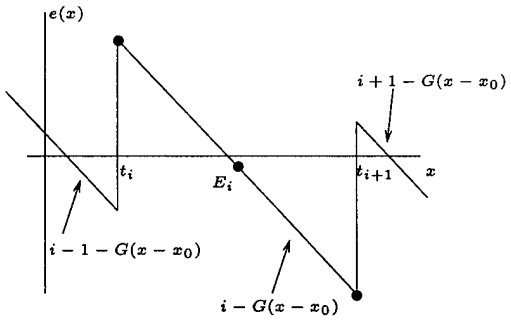

Figure 2: Static Error Description

Fig. 1 shows a non-ideal transfer function relating input voltage to ADC output codes. G is the nominal gain of the converter and the t_i's are quantization thresholds for the ADC. Quantization thresholds determine input voltage ranges that correspond to specific ADC output codes. Fig. 2 is constructed by subtracting the nominal characteristic, $G(x-x_o)$, from the transfer function and zooming in on the ith output state. This figure shows the error for state i and the average error for this state is referred to as expected error, E_i. The expected error for state i is described by

$$E_i = i - G\left(\frac{t_i + t_{i+1}}{2} - x_o\right). \quad i = 1, 2, \ldots, 2^n - 2. \quad (1)$$

G and x_o are the gain and offset, n is the number of bits, and t_i, t_{i+1} are quantization thresholds that yield the ith output state.

III. Sinewave Histogram Methods

The sinewave histogram method [1] is used to estimate quantization thresholds,t_i, for the ADC driven by a single sinusoidal input. Estimated thresholds are obtained from (2).

$$t_i = -A\cos\left(\pi\frac{s_{i-1}}{N}\right) + c. \quad i = 1, 2, \ldots, 2^n - 1. \quad (2)$$

t_i is the transition threshold between states $i-1$ and i, A and c are the amplitude and DC offset of the input sinusoid, s_{i-1} is the number of times that the ADC output codes 0 to $i-1$ occurred, and N is the number of samples in the set. A description for the estimated expected error of a particular ADC output code is obtained by combining (1) with (2) which yields (3).

$$\hat{E}_i = i + GA\left(\frac{\cos\left(\pi\frac{s_{i-1}}{N}\right) + \cos\left(\pi\frac{s_i}{N}\right)}{2}\right) - G(c - x_o). \quad (3)$$

Equation (3) contains two unknown parameters GA and $G(c-x_o)$. The input-voltage-to-ADC-output-code transfer function parameters, G and x_o, for a single input sinusoid are not known exactly. The exact amplitude, A, and DC offset, c, of the input sinusoid is also not known. In addition, both the converter circuitry and the test setup can introduce gain and offset error to the input signal of the ADC. A number of techniques exist to assign values to gain and offset errors [1]. The independent-based method is used in this paper [1]. This method solves for the two unknowns, GA and $G(c-x_o)$, that minimize the mean-square expected error over all states.

An important characteristic of expected error for an ADC must be discussed. It can be shown that the expected error for an ADC depends on the dynamic behavior of the input signal. Fig. 3 shows estimated expected errors for the full range of an 8-bit ADC for a single sinusoidal input. The samples were divided into two sets based upon the polarity of the slope of the sampled signal. The sorting is achieved by obtaining a slope estimate associated with each sample and separating the samples into positive and negative sloped sample sets. Using the terminal-based approach [1] to estimate nominal characteristics, (3) is used to obtain expected errors for each sample set. From Fig. 3 it is evident that estimated expected error follows a different pattern for the increasing sample set than that of the decreasing sample set.

IV. An ADC Error Model

Dynamic performance is described as the behavior of the ADC when time varying signals are present at the input

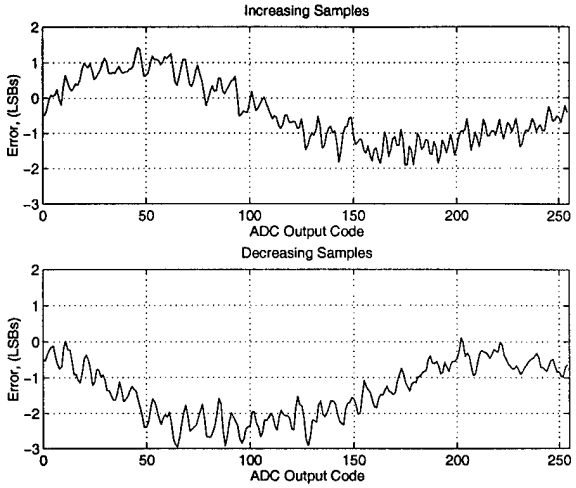

Figure 3: Estimated Expected Error for Increasing and Decreasing Samples

rather than static signals. Parameters used in this paper to describe dynamic error characteristics are the slopes of the input signal and the corresponding output code states. This error is described by

$$\hat{e}_i(y) = \sum_{k=1}^{M} \alpha_{i,k} b_k(y). \quad (4)$$

$\hat{e}_i(y)$ is estimated expected error at state i evaluated at slope y, $b_k(y)$ is the k_{th} basis function evaluated at slope y, and $\alpha_{i,k}$ is the k_{th} basis function coefficient for state i. There are $2^{(\#\ of\ ADC\ bits)}$ sets of basis function coefficients for all states that need to be represented. Equation (4) yields an error function of slope for every state of the converter. Practical converters exhibit discontinuous error-versus-slope patterns from state-to-state. Allowing an error-versus-slope function for every state increases the resolution of this compensation routine relative to previous compensation methods [2].

A variety of basis functions can be used to describe the dynamic error in (4) but there can be problems in the choice of basis functions to use. The estimated error is usually on the order of a few least significant bits (LSBs) and the slope y is many degrees of magnitude larger. Usually a scale factor is placed on the slope variable to achieve better numeric stability in the estimation procedures for $\alpha_{i,k}$.

V. Calibrating the ADC

Calibration of the ADC involves the selection of the model parameters, $\alpha_{i,k}$ in (4), to predict expected errors for the ADC. Sinewave histogram techniques are used to accomplish this task. To accurately describe error, the ADC must

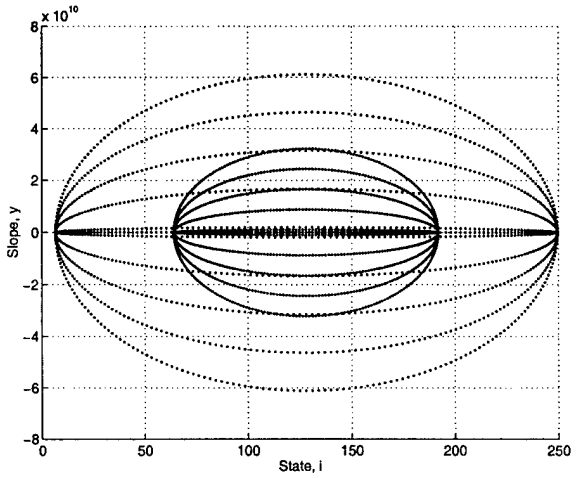

Figure 4: Trajectory Plot using 5 Frequencies at 2 Amplitudes per Frequency

be excited over its full range of state and slope values. Error observed in response to applied test signals determine appropriate coefficients for the basis functions used in (4).

A. State and Slope Space

Each sinusoidal test signal generates an elliptical trajectory in state and slope space and choosing various frequencies and amplitudes yields a whole series of trajectories as shown in Fig. 4. These sets are used to obtain a best fit of (4) to (3). A large number of samples are collected for each sinusoid input to the ADC. (Results shown here all use 64k sample sets.) Slope estimates are obtained from the samples and separate histograms are constructed based upon positive and negative slopes to form estimates of the expected error for the upper and lower half of each trajectory. Each trajectory introduces two unknowns GA and $G(c - x_o)$ as shown in (3).

B. Estimation of Basis Function Coefficients

The following outlines the procedure developed to achieve the required optimal estimates for all $\alpha_{i,k}$. A complete description of the required mathematics may be found in [3]. A conventional way of estimating the unknowns in (3) is to use the FFT of the samples to obtain estimates for c and A and to use a terminal or independent-based method [1] to estimate G and x_o. This procedure could be used individually for each trajectory and the resulting estimated parameters could be used to calculate estimated expected error from (3). Every ADC output state i excited by a trajectory has two slope and error terms associated with it. For state i the basis functions of (4) could then be used to "best" fit a curve versus slope to the expected error of (3). The result of this procedure gives a description of expected error based on state and slope of the input. However, experiments have shown that this method for finding c, A, G, and x_o individually for each trajectory causes a severe degradation in performance of the compensation scheme. The degradation is due to the fact that G and x_0 are global parameters of the ADC and need to be estimated over the entire dynamic range of state and slope, rather than just for a single trajectory. A different approach must be used to estimate the unknown parameters.

The goal of this compensation scheme is to estimate expected error as a function of slope for each state of the ADC. In order to calculate estimated expected error we need to solve for GA and $G(c - x_o)$ in (3). This is accomplished by first finding GA and $G(c - x_0)$ for one arbitrarily selected reference trajectory. This is implemented using the independent-based method [1]. By setting (3) equal to (4) for all states and finding a least square solution for the basis coefficients (α's), an expression is obtained for the basis coefficients for all states as a function of the GA and $G(c - x_o)$ terms in (3) for all trajectories. A least square solution is then obtained to minimize the model error over all values of GA and $G(c - x_o)$. The minimization is performed under the constraint that the unknowns found for the reference trajectory remain fixed. The result is an expression which yields GA and $G(c - x_o)$ for all the trajectories. Using these estimates, the $\alpha_{i,k}$ terms of (3) may be obtained. These values, in turn are used to calculate the error characteristic as a function of state and slope.

VI. Compensating Samples

A method for obtaining a description of error as a function of input state and slope for a particular ADC has been developed. Compensating an output sample set of the ADC is straightforward. Obtain a sample set from the ADC and a slope estimate of the sample set. Take each element of the sample set along with its associated slope value and evaluate (4) to find the estimated expected error value. Subtract this error from the current element of the sample set. The result is a compensated sample set.

VII. Results

Figure 5 shows uncompensated and compensated frequency spectra for a simulated 8-bit folding ADC [4]. The graphs illustrate the potential performance of this compensation technique. No other dynamic compensation techniques have provided the ability to remove high order harmonic distortion from such a complex model. This method clearly removes all distortion to the noise floor. Figure 6 shows a spurious free dynamic range (SFDR) result for both compensated and uncompensated data for a real 8-bit Flash ADC operated at 204.8 MSPS. The SFDR is a dynamic range test that measures the difference (in dB) between the fundamental and highest distortion compo-

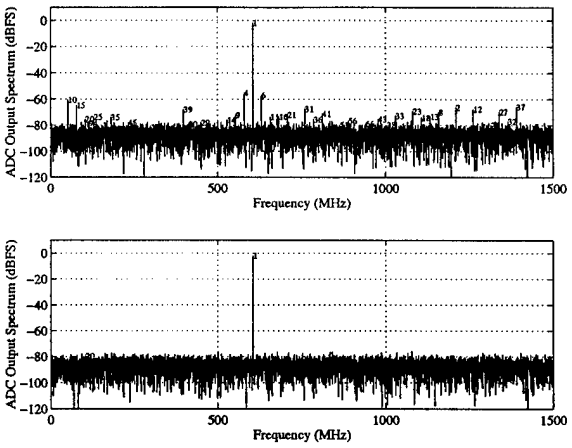

Figure 5: Uncompensated and Compensated Frequency Spectra, Simulated Results

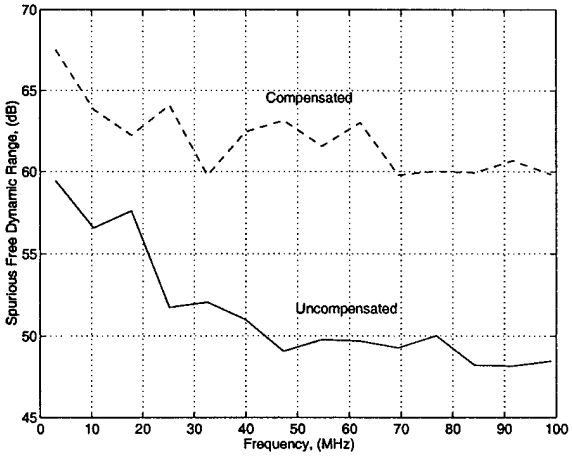

Figure 6: SFDR Plot, Experimental Results

nent over the full Nyquist band for a constant amplitude sine wave inputs. The curve shows excellent improvement in performance across the full Nyquist band. To date this method has provided superior results compared to methods previously reported [5, 2].

VIII. Conclusion

The error introduced by the simulated 8-bit folding ADC used to obtain the results shown in Fig. 5 depended only upon state and slope of the input signal. The results thus show that this compensation method is highly effective for removing all errors that depend on state and slope. The results of running this method on real ADCs are not as profound. However, broadband improvements have been obtained. The exact mechanisms that create ADC error are not always known and often rely on more than just current state and slope. By studying residual errors which remain after compensation, it will be possible to deduce other variable dependencies for the remaining errors.

References

[1] Institute of Electrical and Electronics Engineers, *IEEE Standard 1057-1994, IEEE Standard for Digitizing Waveform Recorders*, Dec. 1994.

[2] D. Hummels, F. Irons, R. Cook, and I. Papantonopoulos, "Characterizaion of ADCs using a non-iterative procedure," in *Proceedings of IEEE International Symp. on Circuits and Systems*, (London), May 1994.

[3] J. Larrabee, "ADC compensation using a sinewave histogram method," Master's thesis, University of Maine, Dept. of Electrical and Computer Eng., Orono Maine, Aug. 1997.

[4] F. Irons, D. Hummels, and C. Zoldi, "ADC architectural diagnostic testing procedures," in *Proceedings of Government MicroCircuit Applications Conference*, (Orlando), Mar. 1996.

[5] F. Irons, D. Hummels, and S. Kennedy, "Improved error compensation for analog-to-digital converters," *IEEE Trans. Circuits and Systems*, vol. 38, pp. 958–961, June 1991.

IEEE Instrumentation and Measurement
Technology Conference
Ottawa, Canada, May 19-21, 1997

A Virtual Instrument Bus Using Network Programming

D.J. Rawnsley, D.M. Hummels, B.E. Segee
University of Maine
Orono, Maine *

Abstract— *This paper provides an overview of a virtual instrument bus created at the University of Maine Orono. Software to support automated tests has become difficult to maintain as the number of test boards and test instruments grows. A variety of test instruments such as logic analyzers, signal generators, and data caches connect and communicate to workstations using a General Purpose Interface Bus (GPIB). This paper describes two software packages. The first is a "virtual instrument bus" that makes a large number of GPIB buses on separate networked computers appear to be on a single bus. The second is an object-oriented instrument library. The Library is designed to support a variety of instruments using a common framework in an easily maintained software package.*

The virtual instrument library is developed using remote procedure calls (RPC). All workstations supporting an instrument bus run a background program called a Bus Server that handles bus communications and provides an interface to the computer network. Communication to the various Bus Servers is handled by the Virtual Bus Library. This interface makes the physical configuration of the instrument buses transparent to the software developer. The virtual bus software provides easy code reuse for quick program generation used for automated testing, at the same time making all instruments appear to be located on one single bus.

I. Introduction

This paper presents the development and implementation of instrument control software for use in a networked computer environment. The project was motivated by ongoing research in the Communication Laboratory at the University of Maine. The Communications Lab, among other things, analyzes Analog to Digital (A/D) converter output to provide a means of compensation for the error introduced by the device. Software to support automated tests for data acquisition from A/D test boards has become difficult to maintain as the number of test boards and test instruments grows. A variety of test instruments such as logic analyzers, signal generators, and data caches connect and communicate to workstations using a General Purpose Interface Bus (GPIB). Software to control test instruments that are physically located on separate workstations within the lab as illustrated in Figure 1 are extremely time consuming or impossible to configure. Moving instruments from one workstation to another required reconfiguring software and recompiling an extensive software package.

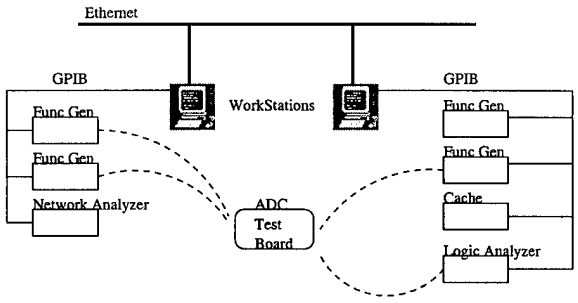

Figure 1: Ideal Lab Configuration

The software maintenance and network support issues encountered on the Communications Lab are typical of those encountered when instruments are controlled over a computer network. While users have become accustomed to distributed network resources (shared file systems, transparent access to printers, etc) instrument control software has not supported the capabilities of most networks. For a networked computing environment, instrument control software should support the following features:

- Instruments should be portable to any machine on the local network without recompiling test software.
- Test software should not be platform dependent. Tests should operate correctly regardless of the platform that the test is run from.
- Development of test software should not be platform

This work has been supported in part by the ARPA HBT/ADC program under a contract administered by the Office of Naval Research Grant N000149311007 and the DEPSCoR program through the Army Research Office Grant DAAH04-94-G-0387

dependent. Once the network instrument control libraries are compiled for a particular architecture, the test software should be supported for any machine using that architecture.

- The software interface should be consistent regardless of the physical instrument bus interface.
- A common software interface should be provided for instruments with common functionality. For example, all function generators should respond to a common set of amplitude/frequency configuration commands.

II. Existing Software

Existing software used in the Lab for data acquisition and controlling instruments is written in the C programming language. The physical addresses of the test instruments and the names of the machine hosts that they are connected to are hard-coded into the software. In order to move instruments from one machine to another, or to change its address, the existing software package has to be recompiled for the changes to take effect.

To access instrument software for a specific instrument, the user has to be logged-on to the workstation to which the instrument is physically connected. Automated tests involving multiple instruments connected to different physical buses cannot be supported. In order to incorporate an instrument on a different bus than the one the test is running on, the cabling would have to be physically changed to the new bus. The address of the instrument would have to be set so that it did not conflict with any other instrument on the new bus location, and the software would have to be recompiled to reflect these changes. Setting up for such changes is time consuming and problematic.

With the expansion of our facility to include new high speed instruments for A/D testing, the current setup is not an efficient use of equipment.

III. The Approach

Two software packages are described which together address these issues. The first is a "virtual instrument bus" which makes a large number of physical buses on a computer network look like a single bus. The Virtual Instrument Library is designed to support the computer network communications making the computer networking transparent to program developers.

The second software package is an object-oriented instrument library which is specific to instruments within the communications lab. The Communications Lab Library is designed to support a variety of instruments using a common framework in an easily maintained software package. The communication between the libraries is illustrated in Figure 2.

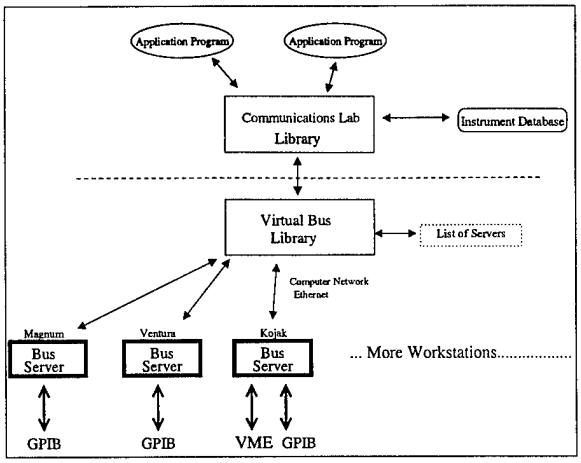

Figure 2: Virtual Bus Block Diagram.

A. Virtual Instrument Library

The virtual instrument library is developed using remote procedure calls (RPC). RPC is a mechanism for building a distributed system of programs that handle all communications between the physical buses, the workstations, and the network. All workstations supporting an instrument bus run a background program that handles all bus communications (like GPIB) and provides an interface to the computer network. These programs are shown in Figure 2 as Bus Servers. The Bus Server can be programmed to communicate with instruments using any kind of bus (not just a GPIB).

Communication to the various Bus Servers is handled by the Virtual Bus Library. This interface makes the physical configuration of the instrument buses transparent to the data acquisition software developer. The library supports a small set of routines modeled after the IEEE 488.2 GPIB standard. It also provides searching functions for locating specific instruments on the computer network, and maintains a list of all machines that have instrument buses connected to them. This library handles all communications to the RPC Bus Servers, and is the interface to the computer network for the Communications Lab library.

B. Communications Lab Library

The Communications Lab library is created using an object-oriented architecture design. It is designed to represent the functionality of the test instruments and provide simplified software reuse and changeability in a modularized fashion. The structure of the library is illustrated in Figure 3.

Object-oriented programming is a method of extending abstract data types to allow for type/subtype relationships among data types. In C++ this is accomplished with inheritance. Instead of re-implementing shared characteris-

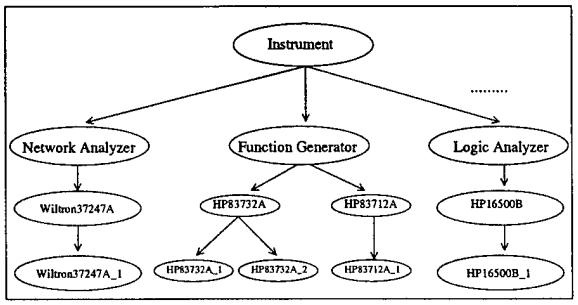

Figure 3: Communications Lab Library Software Structure.

tics, an object can inherit the functionality of the class it was derived from. The C++ class mechanism allows programmers to define their own data type.

The Communications Lab library uses C++ inheritance extensively. Each level in Figure 3 inherits the functionality of the level above it. All test equipment are bus instruments that have common `Instrument` functions. A piece of test equipment, such as a `Function Generator`, has its own common functions to complement the common `Instrument` functions. For example, every `Function Generator` supports a common software interface for controlling the frequency or amplitude of the generator. A specific generator is a Hewlett Packard 83732a which has a variety of functions that are provided which are specific to that model.

The Communications Lab library maintains a bus configuration database shown in Figure 2 which is automatically updated if a change to an instrument's address or location is detected. When one of these instruments, like an `HP83732a`, is used in a program, the library first checks the current location and address in the instrument database. This is done to make sure the program is talking to the correct instrument. If not, search functions of the Virtual Bus library are run to locate the test instrument and update the instrument database of the new test instrument location and address.

The virtual bus software provides easy code reuse for quick program generation used for automated testing, at the same time making all instruments appear to be located on one single bus. This software will greatly facilitate the future development of complex experiments requiring multiple bus instrument coordination.

IV. Virtual Bus Software Architecture

This section gives a quick overview of the software architecture including the names and purposes of the major executables and routines. Figure 4 shows the client-server architecture used for the Virtual Bus Software.

A. Client Side

The Application Programs are the client side of the architecture. All Application programs use the two software libraries, the Communications Lab Library and the Virtual Bus Library, to create client executables. The Communications Lab Library is an object-oriented library that models types of instruments, and communicates with the Instrument Database Server for up to date information on instrument locations. The Virtual Bus Library is the interface to the network communications. This interface is used by the Communications Lab Library to provide reusable objects for Application Programs.

A.1. Virtual Bus Interface

The interface for the virtual bus abstracts away the ideas of network programming from the Communications Lab Library and Application Programs. All interface functions establish connections with the specified servers and handle network communications. When completed, each routine disconnects from the server. Each routine provides an interface that makes it appear that the routine is running locally. When in fact, it maybe executing on a different workstation. The following is brief review of each interface routine.

1. **v_send()**: Send commands or data to a specified instrument.
2. **v_receive()**: Receive data from a specified instrument.
3. **v_bustimeout()**: Set the timeout value for the physical bus. The timeout value is the approximate minimum length of time that I/O functions can take before a timeout occurs.
4. **v_findlisteners()**: Poll the bus to find the number of listeners.

There are two helper functions that are used by `v_findlisteners()`:

1. **get_valid_addresses()**: Build a list of addresses for the `v_findlisteners()` function.
2. **gethosts()**: Get a list of host workstations and possible bus addresses from a configuration file.

B. Server Side

Two different types of servers are used for the virtual bus: the Instrument Server and the Instrument Database Server.

B.1. Instrument Server

The Instrument Server, also called the Bus Server, is run as a background process which is configured by the

`startgpibd` executable. When this process is started during workstation boot-up, it is replaced with the `gpibd` executable. `gpibd` is the server that handles all client requests to communicate with the instrument bus. When a connection is made, a specific service is performed by calling one of the following routines:

1. **v_send_1()**: Send commands or data to a specified instrument physically connected to the same workstation this procedure is executed on.
2. **v_receive_1()**: Receive data from a specified instrument physically connected to the same workstation this procedure is executed on.
3. **v_bustimeout_1()**: Sets the timeout value for the local bus.
4. **v_findlisteners_1()**: Poll the local bus to find the number of listeners.

Each one of these routines calls vender specific GPIB interface software to communicate on the bus.

B.2. Instrument Database Server

The Instrument Database Server is run as a background process which is configured by the `startcommd` executable. When this process is started during workstation boot-up, it is replaced with the `commd` executable. The `commd` server handles all client requests for information about the location of a specific instrument. This server provides two database services:

1. **locate_1()**: Given an instrument identifier, return the last known location of that instrument.
2. **update_1()**: Update the location of an instrument in the database to the current location.

There maybe as many Instrument Servers as there are workstations that have external buses, but only one Instrument Database Server is needed to maintain instrument locations.

V. Conclusions

The Virtual Instrument Bus software has proven to be an excellent software package for data acquisition across a local network. The convenience of running and creating data acquisition software from any workstation on the network makes development easy for the user. The ease of moving instrument locations and changing instrument addresses for specific test setups without recompiling software allows for easy configuration of automated tests. Once an instrument has had its location or address changed the software will update the database so that no searching will take place the next time the software is run. The Virtual

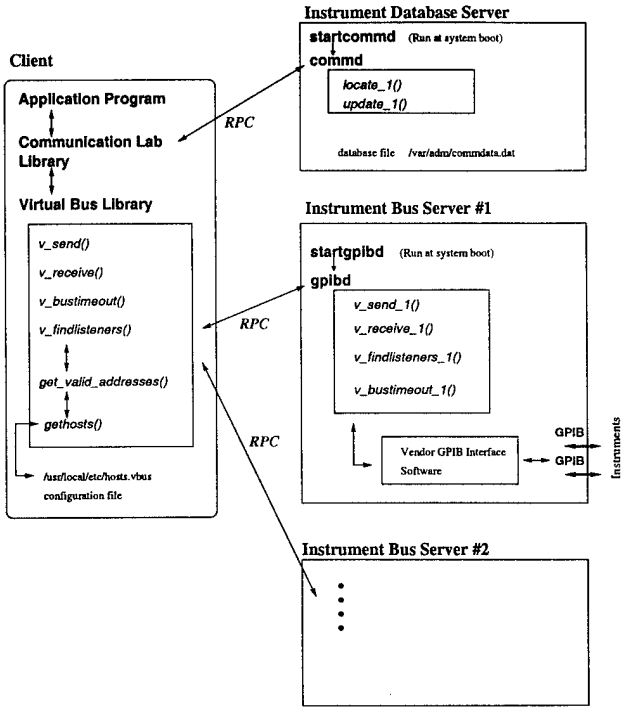

Figure 4: Software Architecture

Instrument Bus software is a powerful tool for providing development of complex experiments requiring multiple bus instrument coordination.

The Modulo Time Plot - A Useful Data Acquisition Diagnostic Tool

Fred H Irons, and Donald M Hummels

Abstract—**This paper illustrates the use of the Modulo Time Plot to facilitate diagnosis of data acquisition systems and components. While conventional techniques, involving spectral analysis and histograms, provide certain useful and necessary measures of performance, the use of reordered sample sets has gained considerable popularity in recent work aimed at characterizing analog-to-digital converter error mechanisms. Examples show that the Modulo Time Plot is useful for quick visual inspection of system performance including dynamic range, distortion and error plots, the detection of random bit errors, and timing errors between the test signal and the sample clock.**

I. Introduction

It has been understood for several years that dynamic testing of analog-to-digital converters (ADCs) and waveform recorders is necessary to fully understand their performance and useful operating parameters [1], [2], [3], [4]. The popular dynamic test signal is the sinewave because it is easy to generate in near ideal form, i.e., with negligible distortion and highly accurate and stable frequency. Normally, when N_S samples are collected and stored in a buffer memory at a sample rate, F_S, the sinewave test frequency is set on one of the basis frequencies, or bins, of the FFT associated with the sample set. This frequency choice eliminates spreading of signal energy across the FFT spectrum that would otherwise occur. The test frequency is also usually chosen to be in the first Nyquist band, i.e., less than $F_S/2$, for most applications; however, increased use of IF detection architectures has pushed testing into second and higher Nyquist bands in many cases.

Whenever a test frequency is not sufficiently less than $F_S/2$, the harmonic distortion, due to the quantizer response to the applied signal, will be aliased across the FFT spectrum. A typical situation is depicted in Figures 1 and 2 for an 8-bit ADC. All the examples presented in this paper will have the following common test parameters, namely: $N_S = 4096$, the clock frequency, $F_S = 204.8$ MSPS, and the test frequency, $F_T = 75.05$ MHz. Figure 1 shows acquired sample values versus sample time in μsec, and Fig. 2 shows the FFT magnitude spectrum versus FFT bin number. Each of the first 50 harmonics

This work has been supported in part by the ARPA HBT/ADC program under a contract administered by the Office of Naval Research Grant N000149311007 and the DEPSCoR program through the Army Research Office Grant DAAH04-94-G-0387

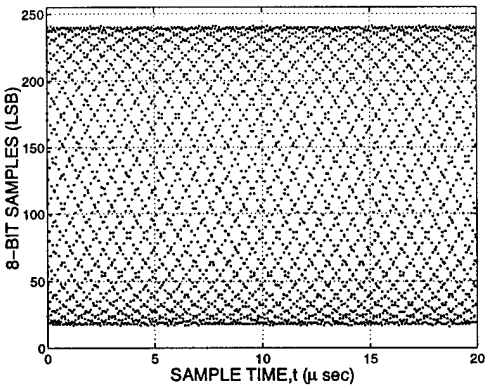

Fig. 1. Sequential time plot of sine wave samples.

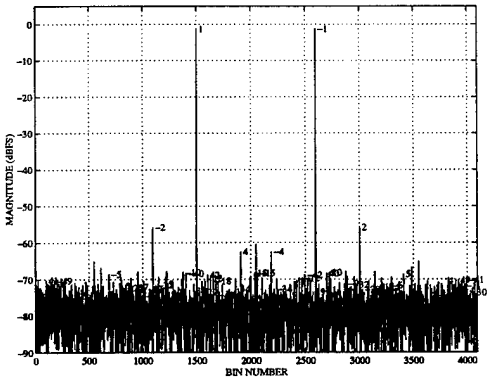

Fig. 2. FFT spectrum for sample of Fig.1.

is labelled if it exceeds a threshold of -75 dB with respect to a full-scale sinewave amplitude. The labeling is (+) for the positive-frequency Euler component and (-) for the conjugate-frequency component. Clearly there are several distortion terms in excess of the noise floor, and clearly the harmonics are distributed across the spectrum by the aliasing process.

As a diagnostic tool, the plot shown in Fig. 1 is not very helpful. It does indicate the maximum and minimum sample values, but aliasing hides the fact that the sampled signal is a sinusoid. The plot provides even less detail when the sample values are connected with straight lines be-

tween each data point as the graph would turn into a solid black bar between the extrema of the set. Consequently, plots of raw samples have not been used for significant diagnostic purposes in the past, other than to construct histograms to observe whether there are any missing or preferred states. This paper shows how a simple reordering of the sample set can be made to yield visually useful information for diagnostic purposes through the use of the Modulo Time Plot.

II. The Modulo Time Plot

The samples shown in Fig. 1 can be rearranged through a straightforward modulo operation which will be developed in this section. The rearrangement usually provides a clearer picture of both the waveform properties and the performance of the sampling device for any periodic test signal.

Let t_k represent the time dependence of the kth sample in the collection of N_S samples at a constant sample frequency, F_S. Then Eqn. 1 represents the set of discrete sample time values based upon the sample number, k, for a set of N_S samples, $x(kT_S)$

$$t_k = k/F_S \qquad \text{for } k = 0, 1, ..., N_S - 1. \quad (1)$$

Equation 1 was used to plot the time axis for the set of sample values shown in Fig. 1. Now consider that when the test frequency is set at some fraction of the sample frequency, the samples are taken at different points in successive periods of the test signal. Let the signal test frequency, F_T, be set on the mth bin of the FFT basis frequency set for the samples as given in Eqn. 2. The number of samples, N_S, is normally a power of 2 for efficient FFT implementation. The m is generally selected to be an odd number for this case

$$F_T = m\frac{F_S}{N_S} \quad \text{for } m \text{ odd.} \quad (2)$$

The period of the test signal is then given by

$$T = \frac{1}{F_T} = \frac{N_S}{mF_S}. \quad (3)$$

When the sample time, t_k, is represented modulo the signal period, T, then the samples may be reordered to display their position within a single period of the test signal. The reordering for plot purposes may be accomplished as follows. Define $\tilde{t}_k$ as the modulo T values of t_k

$$\tilde{t}_k = t_k \; mod \, T. \quad (4)$$

A plot of $x(kT_S)$ versus $\tilde{t}_k$ will be referred to as the Modulo Time Plot. It is possible to show that if N_S is a power of 2, and m is odd, i.e., m and N_S are relatively prime, then the values of $\tilde{t}_k$ are uniformly spaced over one full period of the test signal. Selecting m and N_S relatively prime may also reduce the experimental uncertainty, as suggested in [5] for the diagnosis of ADC integral and differential nonlinearities. In this case, however, reordering is the issue so that the modulo time plot may also be created by reordering the samples, $x(kT_S)$, to relate index numbers of the original set to indices of a reordered set

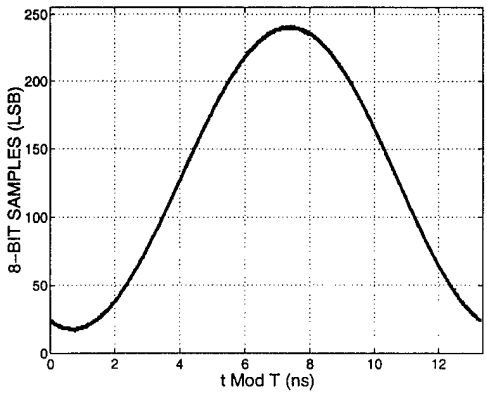

Fig. 3. Modulo time plot for samples of Fig.1.

$$k_R = mk \bmod N_S \equiv mF_S t_k \bmod mF_S T \; . \quad (5)$$

The k_R gives the index of the kth sample, $x(kT_S)$, in the reordered set. The reordered set variables are then given by

$$\begin{aligned} x_R(k_R) &= x(k) \equiv x(kT_S) \\ t_R &= \frac{k}{N_S F_T} \equiv \frac{k}{mF_S} \; . \end{aligned} \quad (6)$$

Figure 3 shows the data of Fig. 1 plotted either as x versus $\tilde{t}_k$ or x_R versus t_R. The samples are plotted as points and are not connected by lines. The reordered set, x_R, can have each value connected with lines; however, the x and $\tilde{t}_k$ points should not be connected when they are plotted. This figure clearly shows that the sampled signal is a sinewave with the correct period and its peak-to-peak operation.

While we have just derived the relationship for the reordered sample set, it is instructive to note that the transform of the reordered set has also reordered the harmonics of the test signal in the following fashion. Equation 7 gives the DFT for the reordered sample set

$$X_R(n) = \sum_{\ell=0}^{N_S-1} x_R(\ell) e^{-j2\pi \ell n / N_S} \; . \quad (7)$$

When N_S and m are relatively prime, Eqn. 7 may be rewritten in terms of a summation index, k, where k and ℓ are related by

$$\ell = mk \bmod N_S \; . \quad (8)$$

Substitution of Eqn. 8 into Eqn. 7 yields the following identity between the transforms for the sample sets

$$\begin{aligned} X_R(n) &= \sum_{k=0}^{N_S-1} x_R(mk \bmod N_S) e^{-j2\pi(mk \bmod N_S)n/N_S} \\ &\equiv \sum_{k=0}^{N_S-1} x(k) e^{-j2\pi mkn/N_S} \\ X_R(n) &\equiv X(nm) \, . \end{aligned} \tag{9}$$

The harmonics of the test signal are reordered sequentially in the transform for the reordered sample set.

The next section presents several examples of how the modulo time plots provide useful time-domain interpretations for sampled data sets.

III. Modulo Time Plot Examples

The Modulo Time Plot is presented as a complimentary tool to assist in the diagnosis of the behavior of a data acquisition system. It is really only one more tool that can be used to determine whether a system is working correctly or not. Examples are given in this section to illustrate the usefulness of this data presentation method.

A. Residual Error

Measured data for a sampled sinewave plus noise is shown in Fig. 1. The noise is added to *dither*, or randomize, what would otherwise be harmonically dependent quantization error [6]. Because a sinewave signal is used, it is possible to estimate both the fundamental component and the average DC component present in the signal. These may be obtained by either an FFT or by methods described in IEEE Standard 1057 [4]. Let A_0 represent the DC estimate and $A_1 \angle \phi_1$ represent the estimated magnitude and phase of the fundamental. A residual error can then be calculated using

$$E(k) = x(k) - A_0 - A_1 cos(2\pi F_T k / F_S + \phi_1) \, . \tag{10}$$

Figure 4 shows the error obtained for the sample set, x, as it appears versus sample time, t. In this pattern everything appears to be fine as the error looks uniform across ± 1 LSB, as it should for a dithered quantizer. There are a few "outliers" that might be expected if the noise source does not have a uniform distribution function.

In spite of this apparently good state of affairs, this residual error actually has a strong correlation to the input signal which is evident when viewed with a Modulo Time Plot. Figure 5 shows the graph of Fig. 4 plotted versus modulo time along with a superimposed and scaled version of the fundamental component of the sample set, x. This plot shows amazing order in the residual error when each point is referred to its correct timing within the test signal period. The adjustment of dither is illustrated at

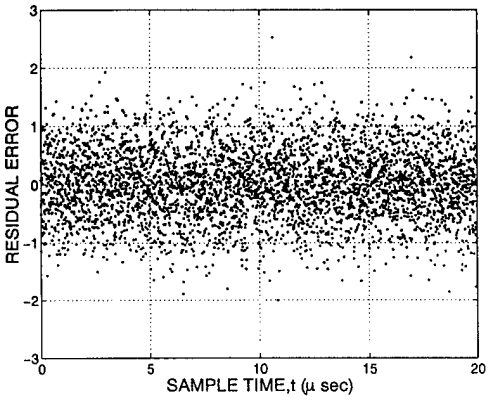

Fig. 4. Sequential time plot of residual error.

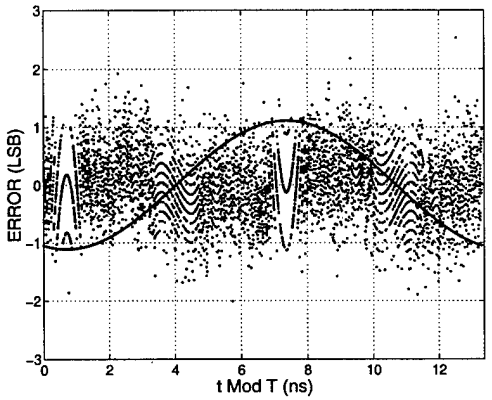

Fig. 5. Modulo time plot of residual error.

the peaks of the sinusoid. Clearly the Modulo Time Plot of Fig. 5 tells us a lot more about what is happening than does the conventional plot of Fig. 4.

B. A Noisy Bit

When one works with high-speed data acquisition it is necessary to electrically match data lines to avoid reflections on transmission lines. A mismatched line causes logical threshold decision errors which in turn affect bit error rates for the transmission of data across digital interfaces. The example shown in this section illustrates the detection of a single noisy bit through the use of the Modulo Time Plot.

Simulated data are obtained for this example by adding random errors to the fourth least significant bit (2^3) of a measured data set so as to simulate a noisy ECL threshold on the data acquisition interface. This causes the bit to have spurious high levels (1's) due to random noise on the data line exceeding some unknown threshold. The contaminated data set is plotted versus time and shown in Fig. 6. Generally, this looks like any other sinusoidal sample set except for a few odd points at the positive peak

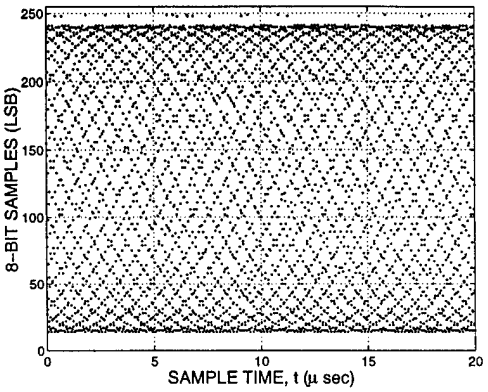

Fig. 6. Sequential time plot of noisy bit samples.

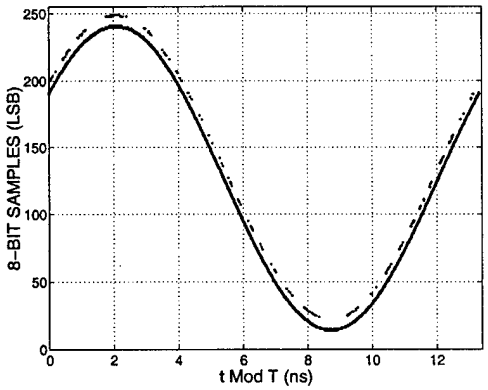

Fig. 7. Modulo time plot of noisy bit samples.

of the data. Inspection of the FFT magnitude spectrum would show a plot nearly identical to that shown in Fig. 2 except that the average value of the noise floor is elevated a few dB. A noisy bit generally does not introduce harmonic distortion; rather, it spreads energy uniformly across the full Nyquist band. The histogram would look very nearly the same as for the ideal sinusoid except there would be a small set of occurrences at 8 LSBs above the usual maximum. Generally the histogram shape is not that helpful for diagnostic purposes.

When the data of Fig. 6 are plotted versus modulo time, the graph shown in Fig. 7 is obtained. Here it is very clear that there is a spurious effect occurring, and it is just a few LSBs above the correct signal. To determine the actual difference it is useful to obtain a residual estimate following the procedure outlined in the previous example. The result is shown in Fig. 8. The residual error shows normal errors as expected over ± 1 LSB, and then spurious errors are clustered around the 8 ± 1 LSB ordinate of the plot. Should there be other noisy bits (due to random effects), there would be points clustered along other power-of-2 ordinates.

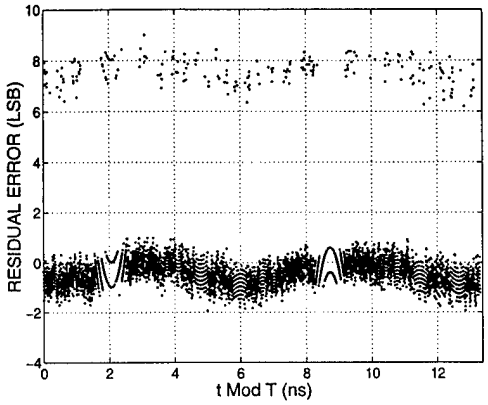

Fig. 8. Modulo time plot of residual error.

C. *Intermittent Sample Clock*

A final example is given for the case where the sample clock is intermittent across the interface between the ADC and the data acquisition system. This type of error could also occur due to improper electrical matching of the line driver for the clock to the data acquisition unit. Whenever a data cache is instructed to collect N_S samples, that is exactly what it does when it is working properly! However, it does not count any missed sample events caused by data line mismatch or noise on the clock enable line. The data presented here were simulated by deleting two samples during the data acquisition cycle.

The sample set is plotted in Fig. 9 versus linear time based upon the nominal sample interval for the experiment. The data look perfectly normal for a sinewave plus noise. However, inspection of the FFT magnitude spectrum would show what appears to be phase noise around the fundamental frequencies of the sinewave. It might be suspected that the test signal is not centered on an FFT basis frequency due to the apparent leakage across the spectrum. The FFT spectrum is not sufficient to tell what is really wrong, and so one can try the Modulo Time Plot as a last resort. The result is shown in Fig. 10. Here the results are very informative. First, the graph tells us that the test signal is set on the correct frequency because we are getting one full-period sinewave response on this plot. The fact that there are three full sinewaves says that the data cache missed two blocks of time during the data acquisition cycle. This example simulated the dropping of only one clock period each time a sample was skipped. The phase shift per clock period is given by the ratio of the test frequency, F_T, to the sample frequency, F_S. For this case we would have a shift of about 132° per clock cycle which appears to be the case for the data shown in Fig. 10. Again, only the data points are plotted, so the density or thickness of each line gives an indication of how long the time intervals were between clock skips in the acquisition process. Once again, the Modulo Time

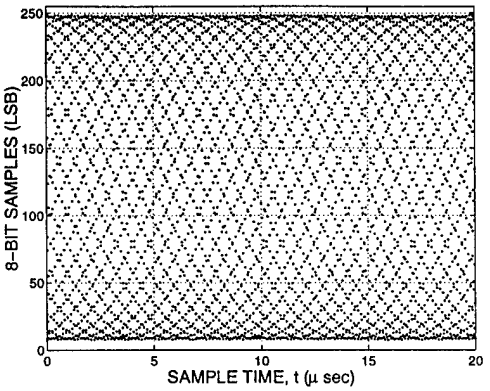

Fig. 9. Sequential time plot of missed clock samples.

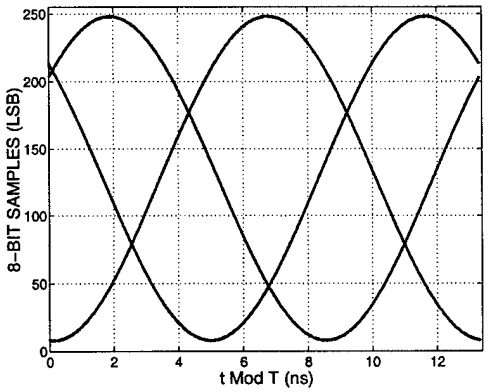

Fig. 10. Modulo time plot of missed clock samples.

Plot helps to figure out what is happening in the sampling process when ordinary time and frequency plots fail to provide a conclusive "picture" of the process.

IV. Conclusions

This paper has demonstrated the usefulness of the Modulo Time Plot as a tool to assist in the diagnosis of data acquisition systems. While modulo arithmetic is not new, nevertheless, the method has not received any real use or been documented for this type of application [4]. The procedure is quite often useful when ordinary time plots are not informative due to aliasing as observed in the time domain, or when spectral and histogram plots do not provide conclusive evidence about what is happening in a process. Three examples of applications have been presented. The first example showed how a residual error estimate had a strong correlation to the fundamental component of the test signal. The second example showed how a noisy bit could be observed from otherwise nearly normal data. The final example illustrated the detection of skipped samples or the effect of intermittent sample clock errors. Each of the examples has similar data when viewed by either spectral analysis plots or plots of the raw data against sample time. Plotting against modulo time brings the data into focus in terms of what is happening in response to any applied periodic test signal.

References

[1] Kester, WA, "Characterizing and Testing A/D and D/A Converters for Color Video Applications," *IEEE Trans on Circuits & Systems*, Vol CAS-25 No.7, Jul 1978, pp 539-549

[2] HP Product Note, "Dynamic Performance Testing of A-to-D Converters," Hewlett Packard, Product Note 5180A-2, 1984

[3] Irons, FH & Rebold, TA, "Characterization of High Frequency Analog to Digital Converters for Spectral Applications," MIT Lincoln Laboratory, Project Report AST-2, Nov 1986

[4] Standard 1057, "Standards for Digitizing Waveform Recorders," *IEEE Standard 1057*, Jul 1989

[5] Vanden Bossche, M, Schoukens, J, & Renneboog, J,"Dynamic Testing and Diagnostics of A/D Converters,"*IEEE Trans on Circuits & Systems*, Vol CAS-33 No.8, Aug 1986, pp 775-785

[6] Jayant, NS, & Rabiner, LR, "The Application of Dither to the Quantization of Speech Signals," *Bell System Technical Journal* 51, 1972, pp 1293-1304

IEEE Instrumentation and Measurement
Technology Conference
Brussels, Belgium, June 4-6, 1996.

ANALOG-TO-DIGITAL CONVERTER ERROR DIAGNOSIS

*FH Irons, DM Hummels, IN Papantonopoulos, and CA Zoldi**
irons@eece.maine.edu
University of Maine, (207) 581-2229

ABSTRACT

Following published procedures for characterizing ADCs using phase-plane error functions, this paper shows how a given calibration data set may used to extract estimates of specific error performance features pertaining to ADC architectural considerations. The procedure requires the selection of basis functions based upon properties of a desired feature. The techniques are applied to the 8-bit TKAD20 operating at 204.8 MSPS to illustrate the concepts discussed in the paper. Results show how it is possible to estimate hysteresis and average sample time errors versus the state of the ADC. A simple consideration shows why it is not possible to separate sample time errors from the effects of nonlinear capacitance and a first ever diagnosis yields sample-time jitter versus ADC state.

1 INTRODUCTION

Previous work[1] has shown that dynamic error representations for an ADC can be obtained directly from a set of sine wave calibration data. The dynamic error is assumed to be a function of two variables, x and y, where x represents the output state and y represents an estimate of the corresponding slope of the state of the ADC output.

$$e(x,y) = \sum_{i=1}^{L} \alpha_i b_i(x,y) \qquad (1)$$

The error function given by (1) is represented over the space defined by x and y for the set of basis functions, b_i. Since y is a measure of $\dot{x}$, the space is often referred to as the phase-plane for the ADC. Once the coefficients, α_i, are determined for each basis function, it is possible to compensate the ADC by removing the error estimate from the data as shown in (2).

$$z_i = x_i - e(x_i, y_i) \qquad (2)$$

In (2), z_i represents the compensated signal where x_i is the i^{th} sample of the ADC and y_i is the slope.

*This work has been supported in part by the ARPA HBT/ADC program under a contract administered by the Office of Naval Research Grant N000149311007

Previous publications have not addressed the issue of what type of basis functions should be used for this problem. Following neural network procedures for the development of training functions, two-dimensional Gaussian functions have been used in the past. These functions have consistently provided well-behaved solutions to the least square procedures used to estimate the ADC error function. The arbitrary use of these functions does not answer the question of whether there are specific functions that will more effectively model ADC error mechanisms. This paper presents some results that have been obtained in a preliminary look at choosing functions based upon different architectural features used in the design of ADCs.

Section 2 presents results obtained using specific basis functions to represent designated error features. In each case the effect of using a specific basis set is evaluated by using the error function to compensate the ADC. The compensation performance is evaluated by measuring the ADC's compensated spurious free dynamic range (SFDR) over the Nyquist band. It should be noted that each of the specific error functions are estimated by using the same calibration data set, thus not requiring any variations or special changes in the calibration circuitry. The results therefore show that a set of calibration data contains all the information necessary to estimate particular error features whenever pertinent basis functions are used.

2 ADC ERROR FEATURES

2.1 Hysteresis

A test used by ADC manufacturers is the measurement of differential nonlinearities. This test is performed by using a computer driven DAC to generate a precision ramp signal. The ramp takes a specified number of steps through each state of the ADC. The ADC is sampled several times at each step with the result that statistics can be assembled for each quantization interval and threshold [2, 3]. This test is virtually a static test except that the results differ for an upward versus a downward ramp. The measurement thus exhibits a hysteresis phenomenon for the quantization threshold parameters.

It is possible to observe the same hysteresis by sampling a pure sine wave signal and constructing an error estimate using the dominant harmonic distortion terms found in the FFT of the sample set. When the error estimate is plot-

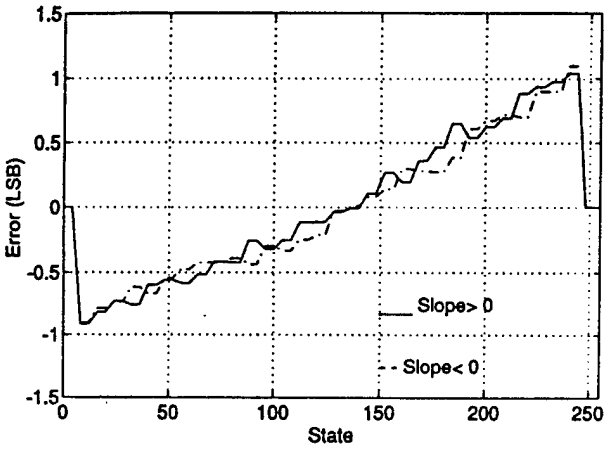

Figure 1. Flash converter dynamic hysteresis estimate

ted versus the state of the ADC it is observed that the error forms an open contour thus illustrating hysteresis as a function of test frequency and ADC state. Hysteresis is a dynamic phenomenon and is measurable by using sinewave data as long as the selected basis functions allow for its representation. One way to allow the presence of hysteresis is shown in (3) using the unit step function.

$$\xi = u(y)f(x) + u(-y)g(x) \tag{3}$$

ξ represents error and is a particular form of basis that can be used in $e(x, y)$ in (1). The error in (3) is given by $f(x)$ when the ADC state is increasing and by $g(x)$ when it is decreasing. Thus (3) allows two distinct error functions based upon the slope of the ADC at any of its output states, x_i.

The model was applied by using 32 unit pulse functions for each of $f(x)$ and $g(x)$. The functions were uniformly centered over the 8-bit range of the TKAD20 (now Maxim MAX100), each with a width of eight states. The resulting 64 coefficients, α_i, were estimated using least square methods [1] on the lowest frequency data of the calibration set; e.g., 2.5 MHz at two amplitudes and with error based upon the first 20 harmonics of each signal. The resulting dynamic hysteresis is shown in Fig.1. The solid curve is error (in LSBs) versus the ADC state for increasing state whereas the dashed curve is for decreasing states. A hysteresis phenomenon is definitely evident for this ADC. The corresponding error is shown as a two dimensional function in Fig.2. Clearly there is no $\dot{x}$ dependence other than the "cut" at $\dot{x} = 0$ where the function switches from $g(x)$ to $f(x)$. The error table of Fig.2 is then used to compensate ADC samples. The performance improvement is illustrated by means of the measured SFDRs shown in Fig.3 where compensated and uncompensated SFDRs are compared. The graph shows that the hysteresis estimate improves low frequency performance by as much as 6 dB out to about 20% of the Nyquist band. This measure clearly shows the extent to which hysteresis is present and should

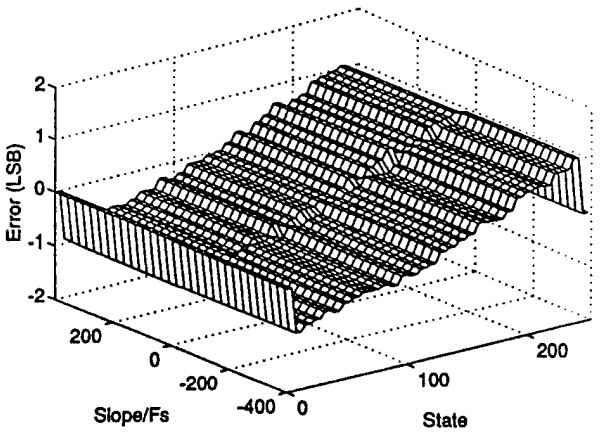

Figure 2. Hysteresis error function in $x - \dot{x}$ space

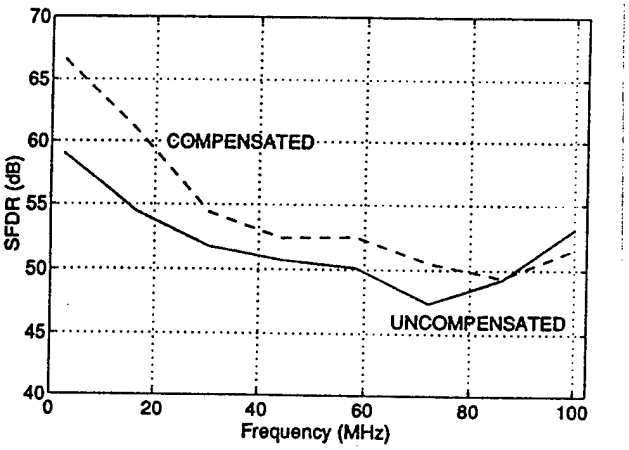

Figure 3. SFDR improvement using hysteresis errors

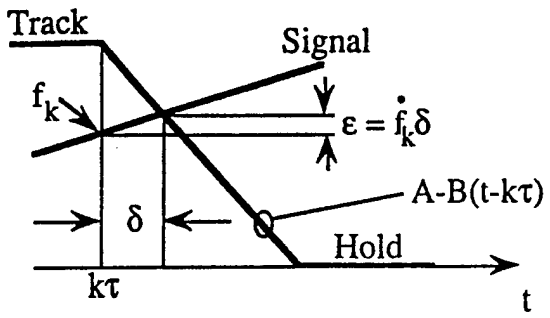

Figure 4. Sample-time error generated by T/H switching

be accounted for in order to improve the dynamic performance of this ADC.

2.2 Sample-time Error

Another important feature that contributes to ADC sample error is referred to as sample-time jitter. Almost all high-speed ADCs use some form of Track/Hold (T/H) circuitry which can contribute to amplitude dependent sample-time errors. Various techniques are used to try and measure this phenomenon and most involve precision filters, phase-locking techniques, and special circuitry to isolate this second order effect. An analysis of the response diagram shown in Fig.4 leads to the following result.

As shown in Fig.4, the state of the ADC is compared to the T/H control signal used to control the switch. Most high-speed sample-holds employ current switching through diode bridge circuits and, as was originally shown by Gray and Kitsopoulas [4], the switch does not change state until some point after the command is initiated. The point is determined by the intersection of the desired signal with the switch transition.

Let x_k = ADC state at the sample time $k\tau$
y_k= corresponding slope
$A - B(t - k\tau)$= Switch transition waveform, then with
δ = sample-time error

we get (4) at the intersection of the two waveforms.

$$x_k + y_k\delta = A - B\delta \tag{4}$$

Rearrangement yields (5).

$$\delta = \frac{A - x_k}{B + y_k} \tag{5}$$

The error is given by δy_k so that (6) is obtained by invoking the fact that $B \gg yk$.

$$\xi = y_k\frac{A - x_k}{B} \tag{6}$$

(6) shows that sample-time errors yield a polynomial (albeit a straight line) in the ADC state variable times the corresponding slope. No higher order terms are involved for a high speed switch.

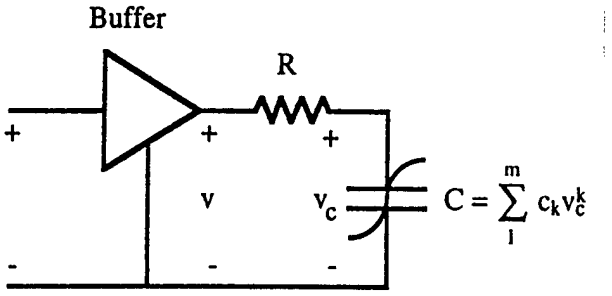

Figure 5. The buffer amplifier drives a nonlinear capacitor

Before testing these error functions it is important to note that a similar result is obtained from any non-linear capacitance that is present in the signal path. Generally, flash comparators yield unavoidable non-linear capacitance at their input. When several (approaching 2^{Nbits}) of these are paralleled, it is difficult to avoid non-linear capacitance on the signal path. As shown in Fig.5, the static capacitance is modeled with a polynomial in the voltage variable, v_C across the capacitor. The voltage variable, v, at the buffer output, is the desired ADC state variable. Thus applying KCL at the capacitor yields (7).

$$\begin{aligned} G(v - v_C) &= G\xi = d(Cv_C)/dt \\ \xi &= R\left(\sum_{k=1}^{m}(k+1)C_k v_C^k\right)\dot{v_C} \end{aligned} \tag{7}$$

The result given in (7) is based on the assumption that ξ is small hence v_C is nearly equal to v and so we obtain the error form given in (8) for a nonlinear capacitor.

$$\xi = yp(x) \tag{8}$$

Nonlinear capacitance in the signal path yields an error basis function which includes the form just derived for amplitude dependent sample-time error.

A measured error function, using the full calibration data set is shown in Fig.6. A fifth order polynomial was used for $p(x)$ and no other basis functions were used to obtain the error table. Since the sample-time error is given by the slope times state, as implied by (6), it is possible to use the error function to obtain an estimate of sample-time error as a function of ADC state. This result is shown in Fig.7.

In the mid-range of ADC values, which corresponds to small incremental signals, the sample-time error appears to have a linear slope thus indicating that the T/H error is dominant for small signals. As the signal is increased, the curve exhibits nonlinear behavior thus suggesting that the nonlinear capacitor dominates sample-time error for large signals. Note that errors ranging from zero to a few picoseconds are obtained from this result without having to resort to any special circuitry or test procedures.

Finally, the estimated sample-time error function is used to compensate the ADC to observe how this error representation affects performance. The result is shown in Fig.8 where compensated and uncompensated SFDRs are compared. The graph shows that the sample-time error provides

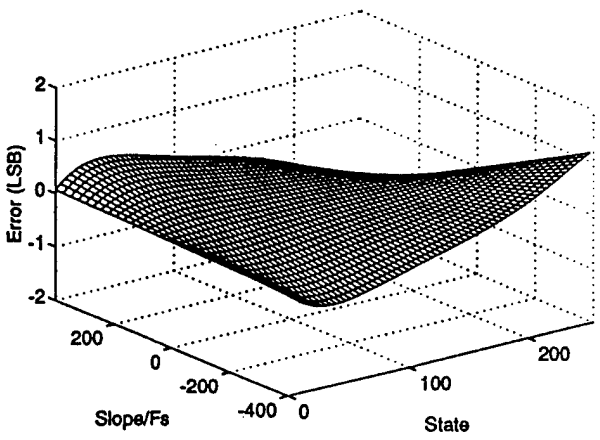

Figure 6. A slope dependent error table

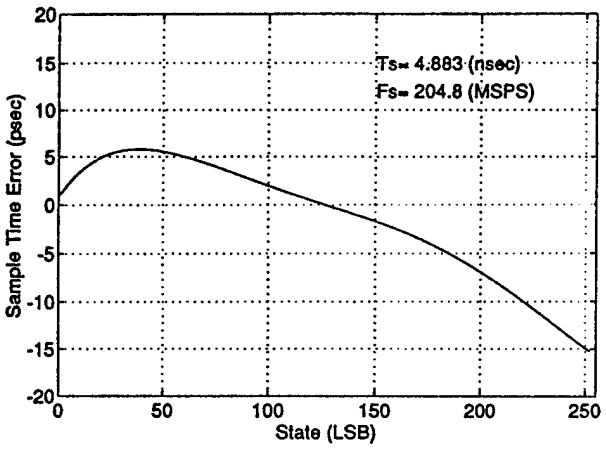

Figure 7. Estimated sample-time error

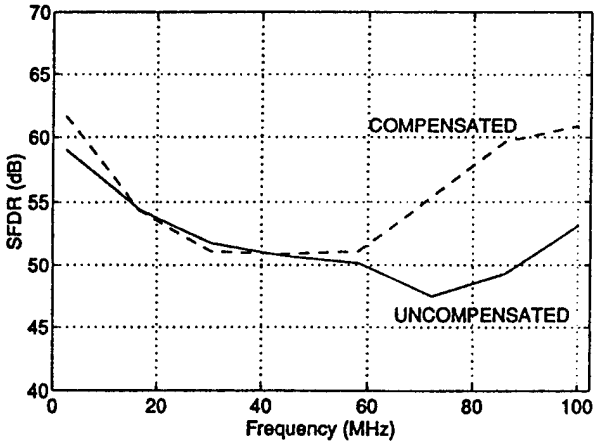

Figure 8. Improvement using sample-time errors

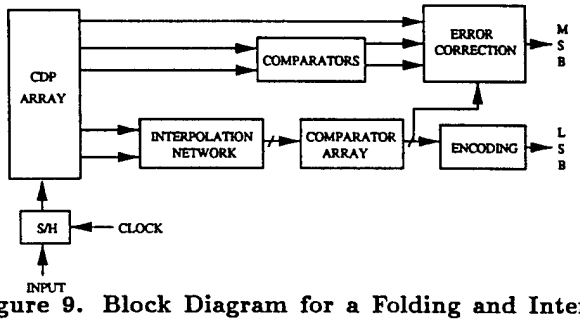

Figure 9. Block Diagram for a Folding and Interpolating Converter.

a significant contribution to dynamic performance for frequencies at the upper end of the Nyquist band. Sample-time error has negligible effect on low frequency performance of the ADC as expected since the parameter, y, goes to zero for low frequency signals.

2.3 Reference Ladder Resistor Values

A final example is used in connection with the development of high speed Folding and Interpolating (FAI) ADCs. These devices are aimed at multi-GHz operation and are on integrated circuit chips that are hard to probe for full bandwidth diagnosis. It is therefor useful to have procedures that estimate specific architectural performance for fully packaged devices. The methods developed here are applied to a simulated FAI ADC and are used to estimate specific resistance variations in a resistor ladder used to set threshold voltages on a Coupled Differential Pair (CDP) array in a FAI architecture.

A block diagram of the simulated FAI 8-bit converter is shown in Figure 9. The converter architecture is modeled after an 8-bit 3 GSPS converter being developed under the ARPA HBT/ADC program. Circuit level simulations of Coupled Differential Pairs (CDPs) were used to form outputs for two folding amplifiers, each with a folding ratio of eight. The amplifier outputs are used to drive a resistive interpolation network, producing a total of 32 folding characteristics. Each of these signals drives a comparator. The comparator outputs are then encoded to determine the five least-significant bits of the converter output. The CDP outputs are used to construct the three most significant bits of the output. A digital error correction scheme is used to ensure that MSB transitions remain aligned with the LSBs.

A variety of error mechanisms could be enabled or disabled through the introduction of non-ideal components into the simulation model. For each mechanism investigated, a "footprint" of the phenomenon can be developed which describes the contribution of each non-ideality to the nonlinearity of the converter. For the FAI architecture, several features have easily identifiable characteristics, so that observed ADC errors can be mapped back into likely causes within the design or manufacture of the device [5]. For example, resistive voltage dividers are used both in the generation of CDP reference voltages, and in interpolation. 22 resistors are used to set reference levels for 22 CDPs that generate quadrature voltage dependent waveforms across the ADC signal (state) space. The ladder is shown in Fig.10.

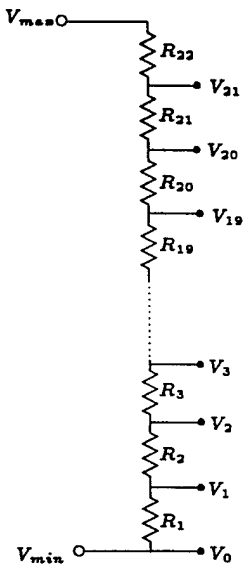

Figure 10. A resistive ladder used for CDP voltage references

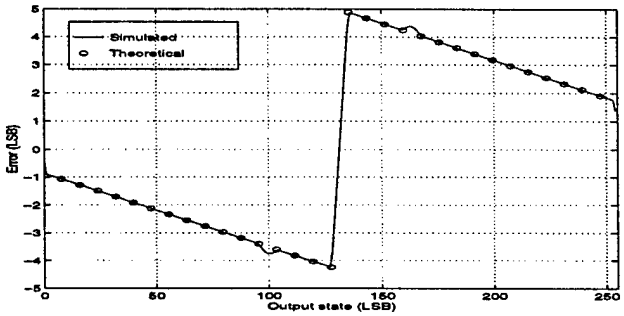

Figure 11. Theoretical and simulated FAI ADC error for a single erroneous resistor in the CDP reference generation network.

An error in a single resistor in the CDP reference voltage network produces an error characteristic as shown in Fig.11. The piece-wise connected characteristic contains two linear regions with the location of the transition depending upon which resistor is in error.

Other error mechanisms are present in the model for the FAI ADC. For a converter with folding ratio 8, an error in a single resistor in the interpolation network affects the converter's output in 8 distinct regions. Similarly, errors due to non-ideal and finite bandwidth CDP circuits, used to model folding amplifiers, are found to be input signal slope dependent. Unlike resistor errors, CDP errors tend to be smooth functions of both the voltage (state) and its time derivative. These errors were all included in the simulated ADC model.

Each resistor within the CDP reference voltage network is identified with a separate basis function, similar to that shown in Fig.11. Sets of simulated FAI ADC samples for sinusoidal inputs were collected and used to estimate ap-

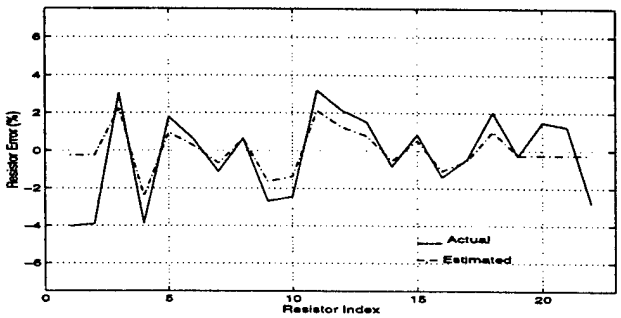

Figure 12. Actual vs Estimated Errors in CDP Reference Resistors

propriate values for α_i in (1) which are then mapped into estimates of specific component error [5]. Fig.12 shows a plot of estimated errors for the 22 resistors used in the CDP threshold generation network. Also shown are the actual errors which were used in the simulation (a 2% tolerance was used to randomize values). The plot shows close agreement between actual and estimated errors for the converter's resistor array.

These results illustrate a powerful procedure for diagnosing sources of distortion introduced by an ADC. Unique error features can be exploited to obtain procedures for deducing internal errors by means of external measures on a device. By isolating flaws, designers are given insight into required design or fabrication process modifications. In addition, external measurements are useful since it is not always practical to perform on-chip dynamic testing of UHF integrated circuit stages or subsections.

3 CONCLUSIONS

This paper has introduced the concept that particular basis functions can be selected to measure specific ADC architectural error phenomena. An 8-bit wide-band flash converter was used to illustrate the estimation of both hysteresis and sample-time errors from a single set of calibration data merely by changing the basis functions used to estimate dynamic error functions. An analysis showed that non-linear capacitance requires a basis function that includes the basis required for the estimation of sample-time errors due to T/H switching. Hysteresis error modeling improved the ADC low frequency performance while sample-time and non-linear capacitor error modeling improved high frequency performance.

Error phenomena considered in this paper did not contribute significantly to midband performance of the ADC. The use of two- dimensional Gaussian, or sinc, functions uniformly distributed over x, y space has historically provided significant improvement for the midband region of the ADC [1]. Work is currently in progress to find alternatives to the algorithms implied by (1) to gain sensitivity to differential error. This extension is required in order to investigate more complicated sources of error, such as differential time-delays in the interpolator output paths for high-speed FAI devices.

REFERENCES

[1] Hummels,DM, et al, "Characterization of ADCs Using a Non-iterative Procedure," *Proc of IEEE Int'l Symp on Circuits & Systems*, London, May 1994

[2] Tewksbury,SK, et al, "Terminology Related to the Performance of S/H, A/D, and D/A Circuits," *IEEE Trans on Circuits & Systems*, Vol CAS-25 No.7, Jul 1978, pp 419-526

[3] Naylor, JR, "Testing Digital/Analog and Analog/Digital Converters," *IEEE Trans on Circuits & Systems*, Vol CAS-25 No.7, Jul 1978, pp 526-538

[4] Gray, JR & Kitsopoulas, SC, "A Precision Sample and Hold Circuit with Subnanosecond Switching," *IEEE Trans on Circuit Theory*, Vol CT-11 No.3, Sep 1964, pp 389-396

[5] Ioannis N. Papantonopoulos, "Error Modeling for Folding and Interpolating Analog to Digital Converters," M.S. Thesis, University of Maine, Orono ME, August 1995.

IEEE Instrumentation and Measurement
Technology Conference
Brussels, Belgium, June 4-6, 1996.

DISTORTION COMPENSATION FOR TIME-INTERLEAVED ANALOG TO DIGITAL CONVERTERS

D.M. Hummels, J.J. McDonald II, F.H. Irons *

Department of Electrical and Computer Engineering
University of Maine, Orono Maine, USA 04469
Phone (207) 581-2245, Fax (207) 581-2220, E-Mail hummels@eece.maine.edu

ABSTRACT

A common technique to acheive high sample rates for analog-to-digital converters (ADCs) is to time interleave two or more devices. A drawback of this approach is that mismatches between the devices cause distortion in the sample sequence. This distortion limits the dynamic range which may be acheived using a particular ADC. Although phase-plane compensation techniques exist to improve the dynamic range of ADCs, these techniques are ineffective for time-interleaved structures. This paper extends the existing phase-plane modeling techniques to time-interleaved architectures. The modified algorithms are tested using a 500 MSPS ADC and are shown to reduce harmonic and intermodulation distortion terms by well over 10 dB.

1. INTRODUCTION

Many high-speed Analog to Digital Converters (ADCs) achieve high sample rates by time interleaving two or more converters. For example, a 1000 MSPS converter might be implemented using two 500 MSPS converters taking samples on alternating clock pulses. One problem with this technique is that mismatches between the two converters introduce distortion into the output sample sequence. For communications applications, this distortion fundamentally limits the potential dynamic range of a receiver which uses time-interleaved ADCs. Dynamic compensation techniques have been developed which use post-processing of the ADC output samples to improve the potential dynamic range [1]. These procedures are capable of reducing the harmonic distortion which is introduced by a single (time-invariant) ADC. However, the techniques are not capable of removing intermodulation of signals with the sampling clock, which is a dominant distortion characteristic for time-interleaved structures. This paper presents a modification of the dynamic compensation algorithm of [1] which may be used to compensate these high-speed structures.

Section 2. illustrates the distortion caused by the time-interleaved structure by presenting measured results for a 500 MSPS ADC. The development of the modified calibration algorithm is given in Section 3.. Section 4. presents experimental results using the new procedure.

*This work has been supported in part by the ARPA HBT/ADC program under a contract administered by the Office of Naval Research Grant N000149311007, the Army Research Office Grant DAAH04-94-G-0387, and by NSF Agency per Grant EEC-9300004.

2. DISTORTION CAUSED BY TIME-INTERLEAVING

The modified algorithms were tested using measured data from a Tektronix AD-10 converter. The AD-10 is an 8-bit 500 MSPS ADC with on-chip track and hold. A block-diagram of the converter is shown in Figure 1. Internally, two 8-bit 250 MSPS converters are time-interleaved to obtain the 500 MSPS sample rate. Each converter has a separate set of individually controllable reference voltages. A typical output spectrum from the converter is shown in Figure 2. The spectrum was obtained using a sampling frequency of $f_s = 512$ MSPS, and driving the converter with a pure sinusoid with test frequency $f_T = 175.125MHz$ (harmonics < 100 dBc). The spectrum shows a variety of spurious signals which are typical of time-interleaved architectures. Signals which are harmonically related to the input signal are labeled by their harmonic number. For this converter, high order harmonics are fairly small relative to the 2nd and 3rd harmonics. The remaining large spurious signals in the spectrum are a result of the time-interleaved architecture. Mismatches between the two converters used to form the sampler modulates the sampled signal at frequency $f_s/2 = 256MHz$. The result is a pair of spurious signals located at $f_s/2 \pm f_T$ (80.875 MHz and 431.125 MHz). The remaining spurious signal located at 256 MHz is due to a DC offset between the two samplers.

Analog adjustments are available to set the reference voltages for the two converters independently, and to adjust the relative phase for the sampling clocks of the converters. Careful adjustment of these voltages can reduce the gain and offset mismatches for the converter. Several factors limit the effectiveness of this approach. First, the gain and phase mismatches are a dynamic phenomenon, and it is difficult or impossible to obtain effective settings over the entire bandwidth of the converter. Secondly, each of the converters is applying a unique non-linear function to its input. Even if the converter gains could be perfectly matched, an intermodulation signal located at $f_s/2 \pm f_T$ would still be present, since the functional form of the non-

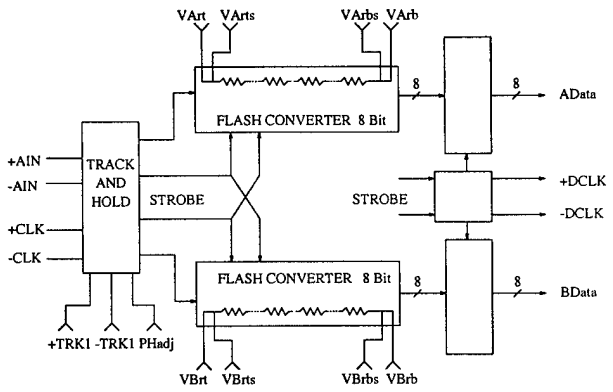

Figure 1. Block Diagram for the Tektronix AD10 Time-interleaved converter

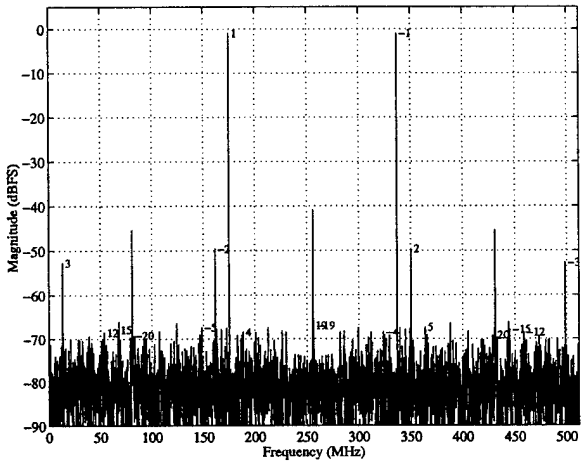

Figure 2. Typical spectra of the output samples from a time-interleaved converter for a pure sinusoidal input.

linearity would alternate on every clock pulse.

3. ALGORITHM DEVELOPMENT

Our goal is to extend the dynamic compensation algorithms of [1] to use post-processing of the digital samples to remove both the harmonic distortion and the intermodulation terms. For a dynamic compensation, the error introduced by the converter for the kth sample is expressed as a function of the converter's output sample, x_k, and the slope of the signal, y_k, at the sample instant. In practice, y_k must be estimated from the data sequence, or measured using additional circuitry. (In this paper y_k is obtained using an FIR filter.) A compensated sample may be written as

$$z_k = x_k - e(x_k, y_k). \tag{1}$$

A calibration algorithm is required to find the function $e(x_k, y_k)$ such that spurious signals are eliminated from the compensated sequence z_k. If the functional form of $e(x_k, y_k)$ does not change with time, then (1) can only influence spurious components which are harmonically related to the input signal. For a time-interleaved converter, the functional form of $e(x_k, y_k)$ must be allowed to change for every other sample (extensions to more than two converters is straightforward).

$$e(x_k, y_k) = \begin{cases} e_E(x_k, y_k) & k \text{ even} \\ e_O(x_k, y_k) & k \text{ odd} \end{cases} \tag{2}$$

A possible calibration procedure is to drive the converter using a sinusoidal test signal, collect N samples of the converter output, and solve for $e(x_k, y_k)$ such that the spurious signals in the compensated sequence z_k are removed. The strength of these spurious signals are given by the DFT of z_k:

$$\sum_{k=0}^{N-1} z_k e^{-j2\pi kn/N} = \sum_{k=0}^{N-1} x_k e^{-j2\pi kn/N} - \sum_{k=0}^{N-1} e(x_k, y_k) e^{-j2\pi kn/N} \tag{3}$$

A flexible form for the error function which allows for efficient numeric evaluation is to use an arbitrary linear combination of a set of fixed basis functions $b_l(x_k, y_k)$.

$$e_E(x_k, y_k) = \sum_{l=1}^{M} \alpha_l b_l(x_k, y_k) \tag{4}$$

$$e_O(x_k, y_k) = \sum_{l=1}^{M} \beta_l b_l(x_k, y_k) \tag{5}$$

The calibration procedure must solve for the appropriate set of coefficients α_l and β_l which remove the spurious components in z_k. Let $n_1, n_2, \ldots n_L$ denote values of n in (3) which correspond to significant spurious terms for a given input signal (for example, 2nd and 3rd harmonics, and $f_s/2 - f_T$). The goal is to find coefficients which make the left side of (3) zero for these values of n. Substituting (4) and (5) into (3), and setting the left side of the equation to zero yields

$$X_{n_i} = \sum_{l=1}^{M} \alpha_l B_{l,n_i}^{(E)} + \sum_{l=1}^{M} \beta_l B_{l,n_i}^{(O)} \qquad i = 1, 2, \ldots, L \tag{6}$$

where

$$X_{n_i} = \sum_{k=0}^{N-1} x_k e^{-j2\pi kn_i/N} \tag{7}$$

$$B_{l,n_i}^{(E)} = \sum_{k \text{ even}} b_l(x_k, y_k) e^{-j2\pi kn_i/N} \tag{8}$$

$$B_{l,n_i}^{(O)} = \sum_{k \text{ odd}} b_l(x_k, y_k) e^{-j2\pi kn_i/N}. \tag{9}$$

Equation (6) gives L complex equations with $2M$ unknowns. Additional equations may be obtained by driving the converter with different calibration test frequencies.

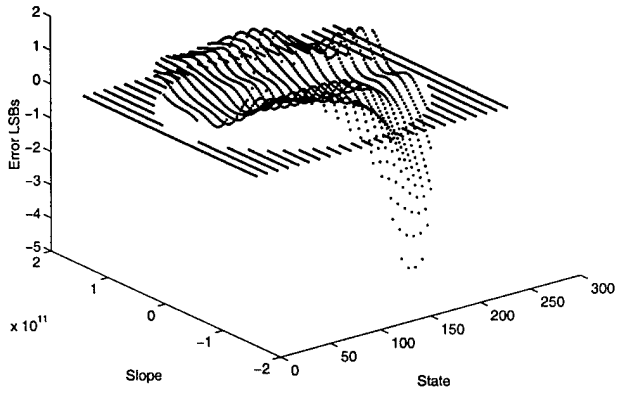

Figure 3. Estimated error characteristic for the even sample converter for the time-interleaved AD10 ADC.

Typically, calibration data is obtained with f_T ranging over the entire band of operation for the device. A large system of equations may be formed in this manner. The equations are solved in the least-squares sense to obtain good values for the unknown coefficients.

4. RESULTS AND CONCLUSIONS

The above calibration procedure was implemented for the AD-10 converter. Ten calibration frequencies from DC to 180 MHz were used. Fifty Gaussian basis functions, centered on equally spaced intervals in the $x_k - y_k$ space, were used to formulate each of the error characteristics. Following calibration, the DC offset between the even and odd sample sets was estimated and subtracted from the odd samples to reduce the $f_s/2$ term of the output sequence. Plots of the error characteristics for the even and odd sample converters are shown in Figures 3 and 4. Clearly the two characteristics are dramatically different from each other, and use of a single correction characteristic for this converter is not appropriate.

A typical spectrum for the compensated sample set is shown in Figure 5. The results show that the above calibration procedure reduces both harmonic and intermodulation terms by well over 10 dB. A measure of the performance of the compensation procedure is the Spurious-Free Dynamic Range (SFDR). The SFDR is defined as the dB difference between the magnitude of the test signal and the magnitude of the largest spurious tone in the full spectrum of the output sample sequence. The compensated and uncompensated SFDR was measured for the AD10 converter using test frequencies ranging over the entire Nyquist band. Figure 6 shows the resulting improvement which was obtained using the time-interleaved compensation scheme.

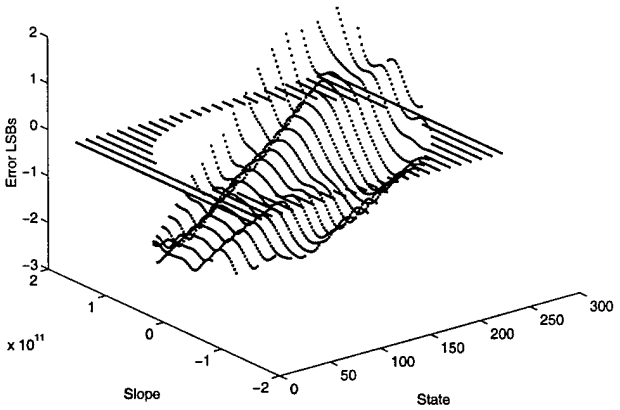

Figure 4. Estimated error characteristic for the odd sample converter for the time-interleaved AD10 ADC.

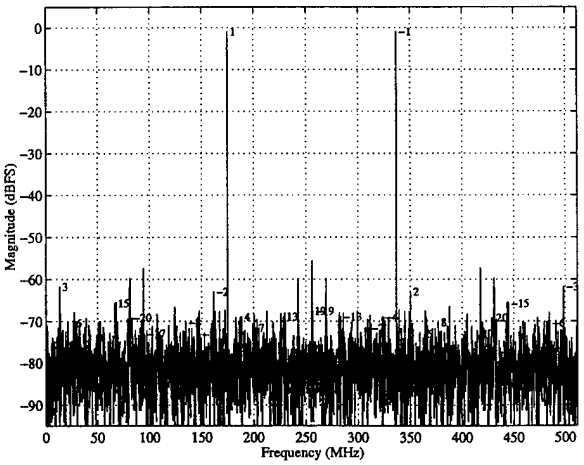

Figure 5. Compensated Spectrum for the AD10 converter.

REFERENCES

[1] D.M. Hummels, F.H. Irons, R. Cook, I.N. Papantonopoulos, "Characterization of ADCs using a non-

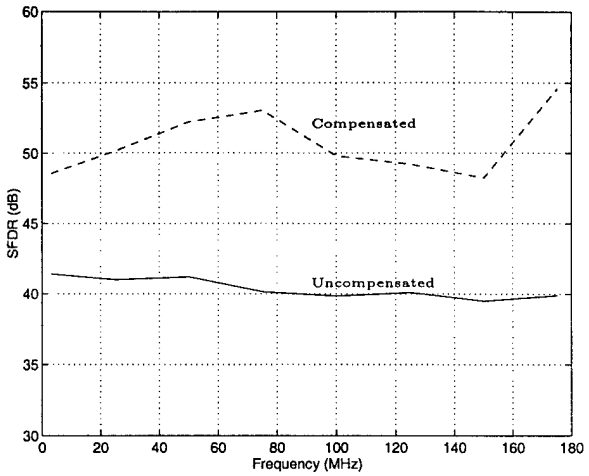

Figure 6. Compensated and uncompensated SFDR for test frequencies covering the Nyquist band.

iterative procedure," Proc. of IEEE International Symposium on Circuits and Systems, London, May 1994.

IDENTIFICATION OF ERROR MECHANISMS IN A FOLDING AND INTERPOLATING ADC

D.M. Hummels, I.N. Papantonopoulos, F.H. Irons *

University of Maine
Orono, Maine, USA
hummels@eece.maine.edu

ABSTRACT

This paper provides an error analysis for Folding and Interpolating (FAI) analog to digital converters, and presents a diagnostic tool for identifying likely causes for measured distortion. FAI converters are efficient since they require a smaller number of components than a classical flash architecture. They reduce both power consumption and die-size, while operating at frequencies comparable to fully parallel architectures. Error mechanisms causing output distortion are identified and explained, and an effective computer simulation module is created that allows error mechanisms to be emphasized or suppressed. For the FAI architecture, various anomalies in the construction of the device are shown to produce specific and identifiable characteristics of the converter's behavior. These characteristics may be exploited in order to diagnose the possible causes of distortion which is observed for a particular device. Techniques are developed which allow measured converter characteristics to be used to isolate flaws, giving designers insight into required design or fabrication process modifications.

1. INTRODUCTION

The concept of signal folding in the implementation of ADCs was first introduced by Arbel and Kurz [1] in 1975. The main motivation was the dramatic reduction of the number of comparators required in the design. Different ways of producing the folding signals have been proposed since then, but the most popular method involves the use of coupled differential pairs (CDPs) [2]. Almost concurrent with the introduction of the CDPs is the concept of resistive interpolation, which produces additional folding signals without requiring additional CDPs.

The diagnostic procedures introduced in this paper rely heavily on the concept of ADC compensation, introduced by Irons and Rebold [3]. Since then, researchers have worked extensively in the field of compensating dynamic errors in high-speed analog-to-digital converters, developing the method of phase plane compensation. This method assumes that the ADC error is mainly a function of the converter's state and the input signal's slope, and focuses on creating an error correction look-up table. Different techniques for obtaining required calibration data have been introduced. Although several methods of compensation have been developed, this paper will rely heavily on the non-iterative ADC characterization procedure introduced in [4], since it allows the use of basis functions to model converter errors. An efficient algorithm that selects the most appropriate basis functions out of a set of possible candidates is also extensively utilized to produce estimates of the converter's error function [5].

*This work has been supported in part by the ARPA HBT/ADC program under a contract administered by the Office of Naval Research Grant N000149311007 and the DEPSCoR program through the Army Research Office Grant DAAH04-94-G-0387

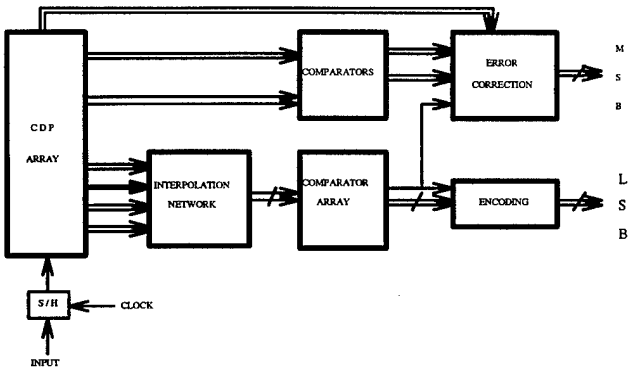

Figure 1. Block Diagram for a Folding and Interpolating Converter.

The technique for performing diagnostics on an analog to digital converter presented in this paper is believed to be novel. Although past schemes have successfully compensated ADCs, this paper introduces the selection of basis functions to model distinct error mechanisms. This allows for the diagnosis of the error mechanisms that contribute to the converter's output distortion.

To verify the performance of the diagnostic techniques, an FAI converter was simulated, so that specific architecture related flaws could be introduced or suppressed. Section 2. describes the particular FAI architecture which was investigated, summarizes the simulation techniques, and describes a variety of anomalies and their effect on the performance of the converter. Section 3. introduces the technique used to isolate the dominant error sources for a packaged FAI converter. Simulation results illustrating the potential of the procedure are presented in Section 4.

2. FAI ERROR CHARACTERISTICS

A block diagram of the simulated FAI 8-bit converter is shown in Figure 1. The heart of the converter is an array of coupled differential pairs. Each CDP is designed to become active over a small range of voltages within the ADC's allowable input range. For this simulation, the outputs of eight CDPs were combined to produce a complementary folding amplifier characteristic which breaks the input voltage range into eight equal-sized regions. By varying the threshold voltages at which the CDPs become active, four

different offset folding characteristics were generated. The set of threshold voltages used in the design of the CDP array are determined by a resistor voltage divider. Each of the four folding amplifier outputs are used to drive a resistive interpolation network, producing a total of 32 offset folding characteristics. Each of these signals drives a comparator. The comparator outputs are then encoded to determine the five least-significant bits of the converter output. The CDP outputs are directly used to construct the three most significant bits of the output word. A digital error correction scheme is used to ensure that the MSB transitions remain aligned with those of the LSBs.

A variety of error mechanisms could be enabled or disabled through the introduction of non-ideal components into the simulation model. For each mechanism investigated, a "footprint" of the phenomenon can be developed which describes the contribution of each non-ideality to the nonlinearity of the converter. For the FAI architecture, many of these characteristics have easily identifiable characteristics, so that observed ADC error characteristics can be mapped back into likely causes within the design or manufacture of the device. In order to isolate the causes of distortion in the output of an FAI converter, it is important to understand the characteristics of the various possible error sources.

Circuit level simulations of Coupled Differential Pairs (CDPs) were used to form dynamic models for the outputs of four folding amplifiers, each with a folding ratio of eight. Transistor parameters were selected which were consistent with a device operating at 3 GSPS. The nonlinear response of a single differential pair was parameterized as a function of the dynamic properties of the input signal. This model was then used to construct a realistic model for the entire CDP array, which was incorporated into the block-diagram of Figure 1. By using ideal values for all components within the FAI converter except for the (non-ideal and finite bandwidth) CDP structures, the contribution to the FAI converter distortion from these structures may be examined. Not surprisingly, the errors due to the CDP circuits used to construct the folding amplifiers may be shown to be input signal slope dependent. These errors tend to be smooth continuous functions of both the input voltage and its derivative.

Another possible source of distortion is non-ideal resistor values. Resistive voltage dividers are used both in the generation of the CDP reference voltages and in the interpolation networks. An error in a single resistor in the CDP reference generation network will result in errors in all of the CDP reference voltages. It can be shown that an error in the m^{th} resistor yields an error in the μ^{th} CDP threshold voltage given by

$$\mathcal{E}_\mu = \begin{cases} (V_{max} - V_{min}) \left[\frac{\mu}{M+\epsilon_m} - \frac{\mu}{M} \right] & \mu < m \\ (V_{max} - V_{min}) \left[\frac{\mu+\epsilon_m}{M+\epsilon_m} - \frac{\mu}{M} \right] & \mu \geq m \end{cases} \quad (1)$$

where V_{max} and V_{min} are the voltage references for the divider, M is the number of resistors, and ϵ_m is the fractional error in the m^{th} resistor. Figure 2 shows a plot of these predicted CDP threshold values, as well as a graph of the simulated converter error using ideal component values for all other components in the converter. The characteristic is composed of piecewise connected linear functions. The transition has a width of 1/32 of full scale and its location depends upon which resistor is in error.

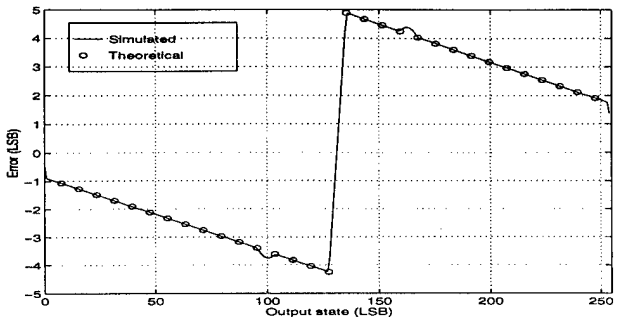

Figure 2. Theoretical and simulated average FAI ADC error for a single erroneous resistor in the CDP reference generation network.

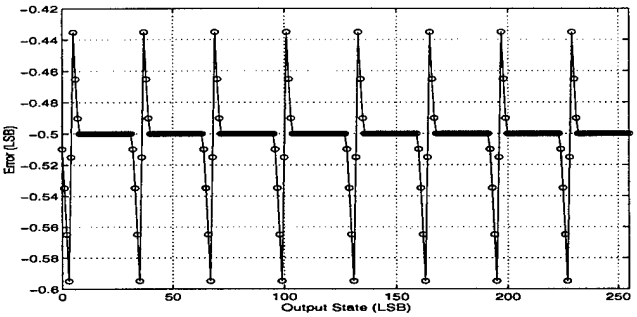

Figure 3. Average FAI ADC Error for a single erroneous resistor in the interpolation network.

An error in an interpolation network resistor results in a very different distortion in the output. For a converter with a folding ratio of 8, an error in a single interpolator resistor influences the converter's output in 8 separate regions. A typical characteristic is shown in Figure 3. The characteristic has the same functional form as that given in (1), except that in this case $(V_{max} - V_{min})$ is one thirty-second of the full-scale range of the converter, and the characteristic is repeated over eight different regions. The location of the discontinuity, and the particular region which is affected is a function of which interpolation resistor (of 32) is in error.

3. ISOLATION OF ERROR CAUSES

The uniqueness of the various error characteristics may be exploited to obtain procedures for deducing the error causes from the behavior of the device. Procedures exist for estimating the error introduced by a device as a linear combination of a set of user-selected basis functions [4]. The formulation has the form

$$e(x,y) = \sum_{i=1}^{N} \alpha_i b_i(x,y), \quad (2)$$

where $e(x,y)$ denotes the error introduced by the converter as a function of the converter input x and the derivative of the input y. $b_i(x,y)$ denotes a basis function, and α_i a coefficient which may be solved for using measured data. The contribution of this paper is the identification of a set of basis functions which model errors introduced by various components of the FAI structure. Under this formulation, the values of α_i may be linked to specific flaws within the device.

For example, for the 8-bit converter simulated in this paper, separate basis function can be selected to model each possible source of error. Errors due to the non-ideal folding amplifier characteristic are smooth functions of both the input voltage and its derivative, and may be modeled using low-order polynomials. Errors due to each resistor within the CDP reference voltage divider may be modeled using a separate basis function given by (1). Basis functions similar to that shown in Figure 3 may be used to reflect the error contribution from each resistor in the interpolation network. The goal is to select appropriate basis functions which reflect possible error mechanisms within the device being tested [6].

Using a set of measured output samples, an error model with the form of Equation (2) may be constructed. In this work, a "Fast Orthogonal Search" (FOS) algorithm was implemented to select the appropriate set of basis functions from the set of candidate functions [5]. Under this procedure, the set of candidate functions is searched to find the basis function which provides the most significant reduction to the squared error of the error model. This basis function is then added to the model. The procedure is repeated until the specified number of basis functions has been selected, or until the model reaches a desired accuracy with respect to the measured data. Using this procedure improves the numerical stability of the algorithm, and provides insight into which basis functions are dominating the error characteristic. The set of selected basis functions, and the corresponding coefficients α_i may then be used to deduce which of the possible error phenomena are dominating the distortion characteristics of the device.

4. SIMULATION RESULTS

The results of this procedure are illustrated in Figures 4 and 5. The simulation was performed for an 8-bit 3 GSPS converter. A collection of output samples for sinusoidal input signals was collected and used to estimate the appropriate values of α_i in equation (2). These values of error basis function coefficients are then mapped into estimates of the specific component errors. Figure 4 shows a plot of the estimated errors for the 44 resistors used in the CDP threshold generation network. Also shown are the actual errors which were used in the simulated converter (a 2% tolerance was placed on the randomized resistor values). The plot shows close agreement between the actual errors and the errors estimated from the converter's output. Similarly, Figure 5 shows the estimated and actual error introduced through the non-ideal behavior of the CDP network used to generate the folding amplifiers. Once again, close agreement was obtained. Errors introduced by the interpolation network were found to be extremely small in comparison to the CDP references and the CDP network. Basis functions associated with this error source were not generally selected by the FOS algorithm as providing a significant source of distortion.

5. CONCLUSION

These results illustrate a powerful new procedure for diagnosing the cause of distortion which is introduced by an ADC. For the FAI architecture, various flaws in the device have been shown to produce specific and identifiable characteristics of the converter's behavior. By matching the functional form of an error model using these characteristics to a set of measured samples from an FAI converter, one may diagnose possible causes of distortion which is observed for

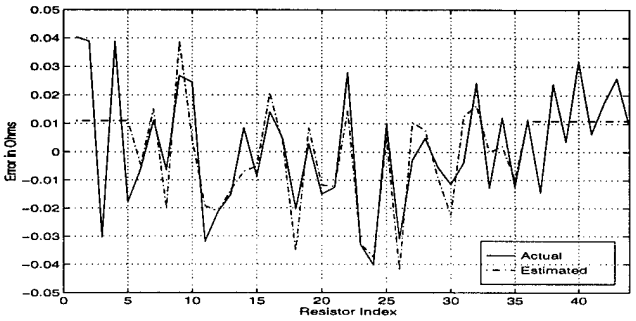

Figure 4. Actual vs Estimated Errors in CDP Reference Resistors

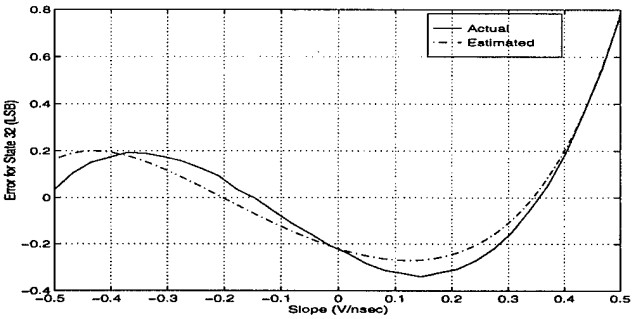

Figure 5. Comparison of Actual and Estimated Errors due to CDP Dynamic Behavior for different signal slopes

a particular device. By isolating flaws, designers are given insight into required design or fabrication process modifications. Work is currently underway to find alternatives to the algorithms of [4] to gain sensitivity to differential error in the converter being tested. This extension is required to investigate more complicated sources of error, such as time-delays in the signal paths for high-speed devices.

REFERENCES

[1] A. Arbel and R. Kurz, "Fast ADC," *IEEE Transactions on Nuclear Science*, vol. NS-22, pp. 446-451, Feb. 1975

[2] Rudy J. van de Plassche and Peter Baltus "An 8-bit 100-MHz Full-Nyquist Analog-to-Digital Converter," *IEEE Journal of Solid-State Circuits*, vol. 23, No. 6, pp. 1334-1344, Dec. 1988.

[3] T.A. Rebold and F.H. Irons, "A phase plane approach to the compensation of high speed analog to digital converters," in *Proc. IEEE Int. Symp. Circuits and Systems*, pp. 455-458, May 1987.

[4] D.M. Hummels, F.H Irons, R. Cook, I. Papantonopoulos, "Characterization of ADCs Using a Non-Iterative Procedure," in *Proc. of Int. Symp. Circuits and Systems (ISCAS '94)* (London, U.K.), vol. 2, pp. 5-8, May 29 - June 2, 1994.

[5] Wahid Ahmed " Fast orthogonal search for training radial basis function neural networks," M.S. Thesis, University of Maine, Orono Maine, Aug. 1994.

[6] Ioannis N. Papantonopoulos, "Error Modeling for Folding and Interpolating Analog to Digital Converters," M.S. Thesis, University of Maine, Orono ME, Aug. 1995.

ADC ARCHITECTURAL DIAGNOSTIC TESTING PROCEDURES

Fred H. Irons, Donald M. Hummels, and Cindy ????? Zoldi *

Department of Electrical and Computer Engineering
University of Maine, Orono, Maine USA
irons@eece.maine.edu

ABSTRACT

Procedures have been developed to apply ADC modeling techniques to the diagnosis of high-speed Folding Amplifier Interpolating Resistive (FAIR) networks. This paper presents results and describes procedures used to obtain measures for specific errors.

1. INTRODUCTION

For some time, it has been known that Analog-to-Digital Converter (ADC) dynamic errors can be modeled using phase-plane functions [1]. This study shows how it is possible to obtain estimates of particular types of errors by using specific basis functions to develop error function models for an ADC. The work presented in this paper is applied to Folding Amplifier Interpolating Resistive (FAIR) networks what were used as basic building blocks for the development of high-speed ADCs under the ARPA HBT/ADC program. Unique aspects of the developed methods are that the procedures are applied to fully packaged ADC devices to estimate internal errors, such as reference resistor values, reference voltage levels, or signal compression. For the FAIR architecture, various anomalies in the construction of the device are shown to produce specific and identifiable characteristics of the converter's behavior. These characteristics may be exploited in order to diagnose the possible causes of distortion which is observed for a particular device.

The methods are tested on simulation models (for which all errors are known). The following sections describe the particular converter architecture examined. The functional form of various potential error sources are then presented. These functions are used to identify the appropriate set of basis functions to describe the error characteristic, providing a means of mapping measured characteristics into possible causes. A series of simulations results illustrating the potential of the approach are then presented.

2. FAIR ADC ERROR SOURCES

A block diagram of the simulated FAI 8-bit converter is shown in Figure 1. The converter architecture is modeled after an 8-bit 3 GSPS converter being developed under the ARPA HBT/ADC program. Circuit level simulations of Coupled Differential Pairs (CDPs) were used to form the outputs of two folding amplifiers, each with a folding ratio of eight. These amplifier outputs are used to drive a resistive interpolation network, producing a total of 32 folding characteristics. Each of these signals drives a comparator. The comparator outputs are then encoded to determine the five least-significant bits of the converter output. The CDP outputs are directly used to construct the three most significant bits of the output word. A digital error correction scheme is used to ensure that the MSB transitions remain aligned with those of the LSBs.

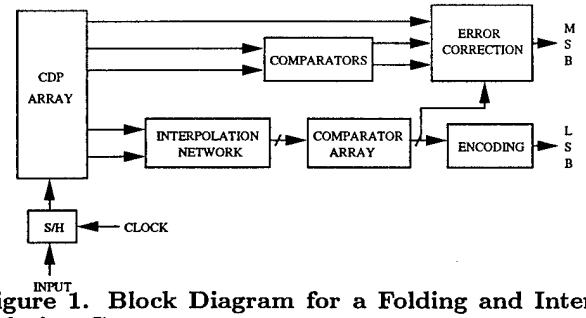

Figure 1. Block Diagram for a Folding and Interpolating Converter.

A variety of error mechanisms could be enabled or disabled through the introduction of non-ideal components into the simulation model. For each mechanism investigated, a "footprint" of the phenomenon can be developed which describes the contribution of each non-ideality to the nonlinearity of the converter. For the FAI architecture, many of these characteristics have easily identifiable characteristics, so that the observed ADC error characteristic to be mapped back into likely causes within the design or manufacture of the device. For example, resistive voltage dividers are used both in the generation of the CDP reference voltages, and in the interpolation network. An error in a single resistor in the CDP reference generation network will be shown to produce an error characteristic as illustrated in Figure 2. The characteristic contains two linear regions connected by a linear transition with width 1/16 of full scale. The location of the transition depends upon which resistor is in error. An error in an interpolation network resistor results in a very different distortion in the output. For a converter with folding ratio 8, an error in a single resistor in the interpolation network may be shown to influence the converter's output in 8 separate regions. A typical characteristic is shown in Figure 3. Similarly, errors due to the (non-ideal and finite bandwidth) CDP structures used to construct the folding amplifiers may be shown to be input signal slope dependent. Unlike the resistor errors, these errors tend to be smooth functions of both the input voltage and its derivative.

*This work has been supported in part by the ARPA HBT/ADC program under a contract administered by the Office of Naval Research Grant N000149311007 and the DEPSCoR program through the Army Research Office Grant DAAH04-94-G-0387

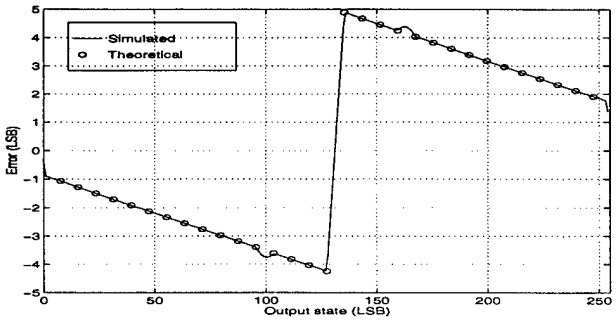

Figure 2. Theoretical and simulated average FAI ADC error for a single erroneous resistor in the CDP reference generation network.

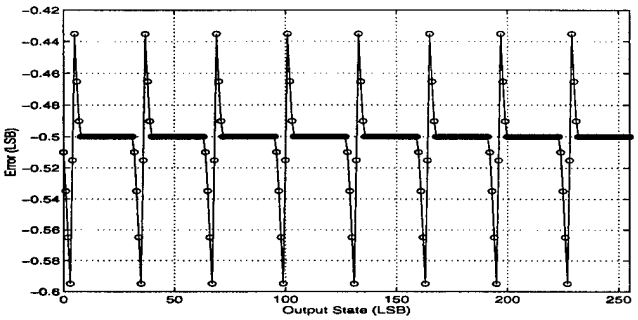

Figure 3. Average FAI ADC Error for a single erroneous resistor in the interpolation network.

3. DEVICE CHARACTERIZATION

The uniqueness of the various error characteristics may be exploited to obtain procedures for deducing the error causes from the behavior of the device. Procedures exist for obtaining accurate models for the error introduced by a device as a linear combination of a set of user-selected basis functions [2]. The formulation has the form

$$e(x,y) = \sum_{i=1}^{N} \alpha_i b_i(x,y), \qquad (1)$$

where $e(x,y)$ denotes the error introduced by the converter as a function of the converter input x and the derivative of the input y. $b_i(x,y)$ denotes a basis function, and α_i a coefficient which may be solved for using measured data. The contribution of this paper is the identification of a set of basis functions which model errors introduced by various components of the FAI structure. Under this formulation, the values of α_i may be linked to specific flaws within the device.

4. RESULTS

The results of this procedure are illustrated in Figures 4 and 5. The results are obtained by associating separate basis functions with individual components of the FAI converter. For example, each resistor within the CDP reference generation network is identified with a separate basis function, similar to that shown in Figure 2. The simulation was performed for an 8-bit 3 GSPS converter. A collection of output samples for sinusoidal input signals was collected and used to estimate the appropriate values of α_i in equation (1). These values of error basis function coefficients

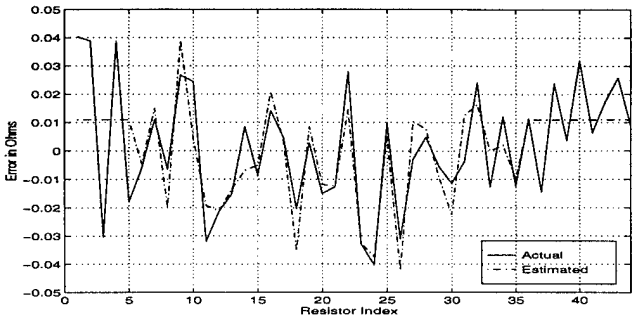

Figure 4. Actual vs Estimated Errors in CDP Reference Resistors

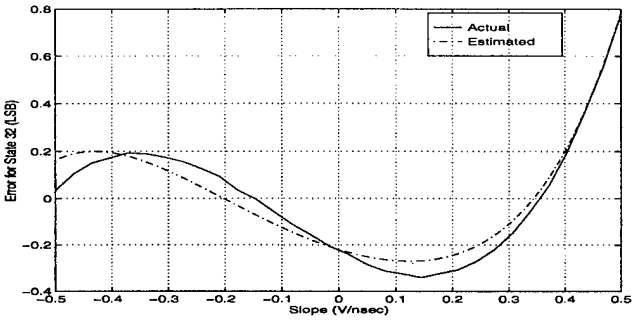

Figure 5. Comparison of Actual and Estimated Errors due to CDP Dynamic Behavior for different signal slopes

are then mapped into estimates of the specific component errors. Figure 4 shows a plot of the estimated errors for the 44 resistors used in the CDP threshold generation network. Also shown are the actual errors which were used in the simulated converter (a 2% tolerance was placed on the randomized resistor values). The plot shows close agreement between the actual errors and the errors estimated from the converter's output. Similarly, Figure 5 shows the estimated and actual error introduced through the non-ideal behavior of the CDP network used to generate the folding amplifiers. Once again, close agreement was obtained. These results illustrate a powerful new procedure for diagnosing the cause of distortion which is introduced by an ADC.

REFERENCES

[1] D. Asta and F.H. Irons, "Dynamic error compensation of analog to digital converters," The Lincoln Laboratory Journal, vol. 2, 1989.

[2] D.M. Hummels, F.H Irons, R. Cook, I. Papantonopoulos, "Characterization of ADCs Using a Non-Iterative Procedure," in *Proc. of Int. Symp. Circuits and Systems (ISCAS '94)* (London, U.K.), vol. 2, pp. 5-8, May 29 - June 2, 1994.

[3] Wahid Ahmed "Fast orthogonal search for training radial basis function neural networks," M.S. Thesis, University of Maine, Orono Maine, August 1994.

Measurement of Random Sample Time Jitter for ADCs

D.M. Hummels, Wahid Ahmed, F.H. Irons*
University of Maine, Orono, Maine
hummels@poirot.eece.maine.edu, (207) 581-2245

Abstract— **This paper addresses the measurement of random sample-time jitter in the characterization of ADC's. A straightforward test is developed which allows for measurement of both additive noise power and RMS sample-time jitter. Simulations are used to assess the accuracy of the technique. Experimental results are also given for a commercially available ADC.**

I. Introduction

Analog-to-Digital Converters (ADC's) are often characterized in terms of the amount of noise which is introduced into the signal during the sampling process. Most ADC manufacturers provide specifications for the number of "effective bits" which the converter is providing. This specification includes the contribution due to a wide variety of noise sources, including thermal and shot noise in the input stages of the converter, errors in the quantization thresholds, and noise which is present on the clock signal.

For many applications, it is desirable to know not only the amount of noise which the converter is introducing, but also the source of the noise. From an ADC designer's point of view, knowledge of whether noise which is introduced through the input signal path, or through instability in the clock signal is critical to focusing the design effort. Similarly, users of ADC's need techniques to isolate noise sources. Converters may exhibit noise levels which are higher than the specifications predict either because of improperly driving the converter, or because of a loss of integrity in the sampling clock.

In this paper, a technique is presented which allows for the separation of additive noise sources from noise which is introduced through sample time jitter. The test is performed by driving the converter using a sinusoidal source, and removing all significant distortion that the converter introduces. The resulting noise process contains quantization noise, thermal noise from the converter input, and noise which is introduced from sample time jitter. In Section II.we show that the sample-time jitter noise has a time-varying variance, since this noise signal is modulated by the derivative of the input signal. A procedure is introduced which measures the power in the modulated process, and relates this quantity to the RMS deviation in the sample time. Simulation results verifying the procedure are presented in Section III. Section IV presents measured results obtained using a commercially available 250 Msps converter.

It should be stressed that variations in the sampling instant which are related to the input signal may also result in distortion. In this paper we do not attempt to characterize the distortion which is introduce by misalignment of the clock signal. Rather, we are concerned with the measurement of the random component of the sample time deviations which contribute to the noise floor of the converter. Measurement of sample-time deviations which result in distortion introduced by the converter is discussed in another paper [1].

II. Formulation

Let $x(t)$ denote the input signal to an ADC. The output of the converter is a sequence of samples y_k, given by

$$y_k = x(kT_s + \Delta_k) + g(x(t))\,|_{t=kT_s} + n_k \qquad (1)$$

In (1), T_s represents the ideal sampling period for the converter, $g()$ represents a nonlinear function to model distortion which is introduced by the converter, and n_k denotes an additive noise component which is due to dithering, quantization noise, and noise sources in the input stage of the converter. The Δ_k term of (1) represents the (random) deviation in the sample time. Our goal is to identify techniques to estimate the variance of the random components Δ_k and n_k, denoted σ_Δ^2 and σ_n^2 respectively. The RMS sample-time jitter is the standard deviation of Δ_k, σ_Δ.

Note that in (1), the distortion function $g()$ may depend not only on $x(t)$ but also on the dynamic properties of $x(t)$ (its derivatives). The magnitude of $g()$ is generally kept

*This work has been supported in part by the ARPA HBT/ADC program under a contract administered by the Office of Naval Research Grant N000149311007

small–on the order of an LSB. Also, manufacturers attempt to control the sampling instant so that Δ_k is small relative to the rate of change of the input signal. Using this fact, y_k may be accurately modeled in terms of the true sample value $x(kT_s)$ and an additive noise component.

$$y_k \approx x(kT_s) + \Delta_k \dot{x}(kT_s) + g(x(t))\,|_{t=kT_s} + n_k \qquad (2)$$

The key to separating the contribution to the noise floor due to Δ_k from that of n_k is to take advantage of the fact that Δ_k is modulated by the derivative of the input signal (errors introduced due to sample-time deviations are largest when the input signal is changing quickly). For measurement of the noise variance, we may drive the converter using a sinusoidal source.

$$\begin{aligned} x(t) &= A\cos(\omega_0 t + \theta) && (3)\\ y_k &= A\cos(\omega_0 kT_s + \theta) + g(x(t))\,|_{t=kT_s} \\ &\quad -A\omega_0\Delta_k \sin(\omega_0 kT_s + \theta) + n_k && (4)\end{aligned}$$

In this case, the distortion function $g()$ is periodic, so that the first line of (4) may be removed from the sample sequence $\{y_k\}$ by finding the FFT of the sequence, and excising all frequencies which are harmonics of the input frequency ω_0. Note that the Δ_k term of (4) is not removed by this process. This term is a random sequence with variance which is periodic at frequency ω_0. The resulting sequence (with periodic components removed) is

$$e_k = -A\omega_0\Delta_k \sin(\omega_0 kT_s + \theta) + n_k \qquad (5)$$

To estimate the variance of the components of (5), we square e_k and evaluate the expected value.

$$\begin{aligned} E\left\{e_k^2\right\} &= E\left\{A^2\omega_0^2\Delta_k^2 \sin^2(\omega_0 kT_s + \theta) + n_k^2\right\} \\ &= E\left\{\left(\frac{A^2\omega_0^2\Delta_k^2}{2} + n_k^2\right)\right. \\ &\quad \left. -\frac{A^2\omega_0^2\Delta_k^2}{2}\cos(2\omega_0 kT_s + 2\theta)\right\} && (6)\\ &= \left(\frac{A^2\omega_0^2\sigma_\Delta^2}{2} + \sigma_n^2\right) \\ &\quad -\frac{A^2\omega_0^2\sigma_\Delta^2}{2}\cos(2\omega_0 kT_s + 2\theta) && (7)\end{aligned}$$

The procedure for estimating the variance σ_Δ^2 is now apparent. The sequence $\{e_k^2\}$ contains a discrete frequency component at twice the test frequency which has magnitude proportional to the desired variance. Once σ_Δ^2 is known, the variance of the additive noise component σ_n^2 may be estimated from the DC component of $\{e_k^2\}$.

The procedure is summarized in the following steps:

1. Drive the converter input with a sinusoidal signal with frequency $\omega_0 = m(\omega_s/N)$, where $\omega_s = 2\pi/T_s$, N is the number of samples collected, and m is an integer. Selection of N as a power of 2, and m as an odd number makes the calculation of the FFT fast, and excites the converter uniformly across the desired states.

2. Collect N samples, $\{y_k : \quad k = 0, 1, \ldots, N-1\}$. Remove the periodic components of $\{y_k\}$ by taking an FFT, removing the DC, fundamental, and all significant harmonics of ω_0, and calculating the inverse tranform of the remainder. The resulting sequence is denoted $\{e_k\}$.

3. Evaluate the FFT of the sequence $\{e_k^2\}$.

$$E_n = \sum_{k=0}^{N-1} e_k^2 e^{-j2\pi nk/N} \qquad (8)$$

   Let C_0 denote the DC term of the signal ($C_0 = E_0/N$), and let C_2 denote the magnitude of the $2\omega_0$ term ($C_2 = 2|E_{2m}|/N$).

4. Calculate the desired estimates:

$$\begin{aligned} \hat{\sigma}_\Delta^2 &= \frac{2C_2}{A^2\omega_0^2} && (9)\\ \hat{\sigma}_n^2 &= C_0 - C_2 && (10)\end{aligned}$$

In step 4, the value of A may be obtained by observing the ω_0 term of the transform taken in step 2. While the theory holds for any input frequency, in practice ω_0 must be chosen large so that the $2\omega_0$ term of step 3 is well above the noise floor. Equation (7) predicts that the strength of the second harmonic term increases quadratically with ω_0.

III. Simulation Results

To test the accuracy of the estimation procedure, a simulation was developed which included a known amount of sample-time jitter. The equations giving the distortion terms of the ADC model are taken from [2]. Let x_1 denote the (dithered) sample $x(kT_s - \Delta_k) + d_k$, where d_k denotes a random dither component which is added to the signal prior to sampling in order to randomize the quantization error. The output of the converter is determined as follows:

$$\begin{aligned} x_2 &= a\tanh\left(\frac{x_1}{b}\right) && (11)\\ x_3 &= x_2 + c\dot{x}_2 - d|x_2|\dot{x}_2 && (12)\\ x_4 &= x_3 + \left[1 - \cosh\left(\frac{x_3}{e}\right)\right] && (13)\\ y_k &= \mathrm{Q}(x_4) && (14)\end{aligned}$$

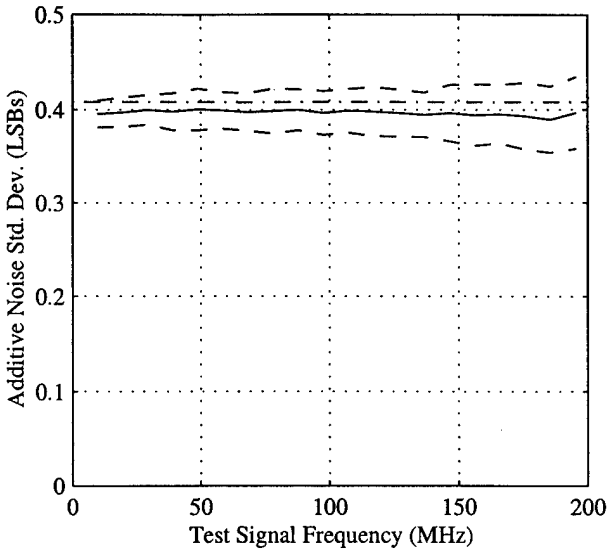

Figure 1: Estimation of σ_n. The solid line indicates the average estimate, and the dotted lines give ± 3 standard deviation bounds. The dash-dot line indicates the actual value of σ_n for the simulated ADC.

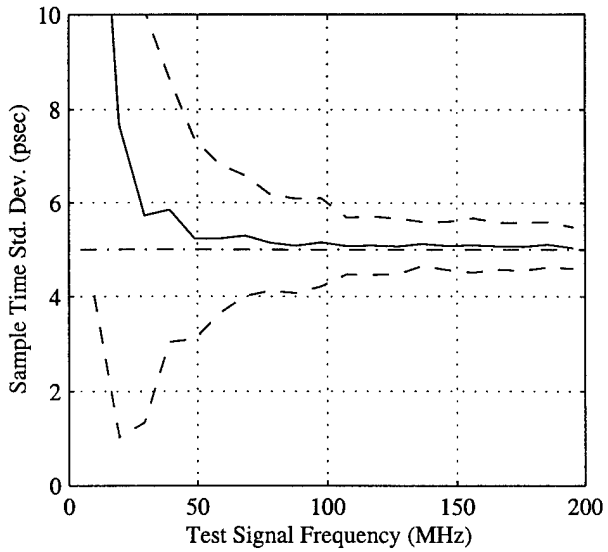

Figure 2: Estimation of σ_Δ. The solid line indicates the average estimate, and the dotted lines give ± 3 standard deviation bounds. The dash-dot line indicates the actual value of σ_Δ for the simulated ADC.

Equations (11) and (13) reflect amplitude distortion in the input and buffer amplifier stages of the converter. Deterministic sample-time offsets result in (12), which is derivable from a symmetrical quad-switching circuit following the analysis of Gray and Kistopoulas [3]. In (14), the function Q() is used to represent quantization to one of 2^n values for an n-bit converter.

All of the results presented here were for a simulated 8-bit converter sampling at 200 MSPS with a peak-to-peak full scale range of 2 V. The simulation parameters used were $a = b = 4$, $c = d = 3 \times 10^{-11}$, and $e = 7.25$. The parameters were chosen to give distortion terms which were roughly consistent with the measured distortion for the 8-bit converter tested in Section IV.. The additive dither d_k was chosen to be a Gaussian random variable with variance equal to the quantization noise power. This gives a theoretical value of $\sigma_n = 1/\sqrt{6} = 0.4082$ LSBs. Δ_k was also chosen to be Gaussian, with standard deviation $\sigma_\Delta = 5$ psec. Test frequencies were varied from 10 MHz to 200 MHz with 50 trials at each frequency. For each trial, 4096 input samples were generated and used to form estimates of σ_n and σ_Δ. Step 2 of the estimation algorithm was implemented by removing the first 20 harmonics of the test signal frequency. Figures 1 and 2 illustrate the results. As expected, the estimate of σ_Δ improves with increasing test signal frequency, as the second harmonic term of (7) comes out of the noise floor. Both estimates appear to be nearly unbiased for large test signal frequencies.

IV. Experimental Results

The estimation procedure was implemented on a Tektronix AD-20 8-bit converter sampling at 204.8 MSPS. Examination of the spectrum (8) showed a significant component at $f_s/2$ in addition to the expected terms at DC and $2\omega_0$. The presence of this term suggested that the converter was actually a time-interleaved converter, employing two converters sampling alternately. To test the observation, the sample sequence was broken into two separate sequences–the first containing the odd numbered samples, and the second containing the even numbered samples. Each of these sample sequences displayed only the DC and $2\omega_0$ terms predicted in Section II.

The estimation algorithm was then implemented on each of these sub-sequences, resulting in two separate estimates of σ_n and σ_Δ. Step 2 of the algorithm was implemented by removing the largest 20 spurious signals in the transform of the sample sequence. Estimates were formed for input signal frequencies ranging from 100 MHz to 200 MHz (No significant $2\omega_0$ term was apparent at frequencies below 100 MHz). Input signal amplitudes were set at 95% of the full-scale range of the converter. In all cases the converter was dithered using Gaussian noise sources with 100 MHz bandwidth and noise power equal to the ideal quantization noise power. This gives a theoretical value for an ideal converter of $\sigma_n = 1/\sqrt{6} = 0.4082$ LSBs. Each estimate was obtained using 16384 samples (8192

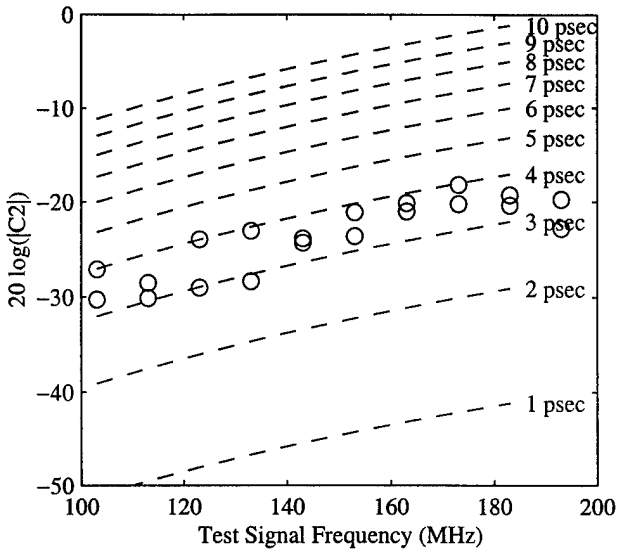

Figure 3: Measured magnitude of C_2 for various test frequencies. Dashed lines show the behavior predicted by the theoretical development for various values of σ_Δ.

samples per sub-sequence).

Figure 3 gives a plot of the second harmonic term C_2 from step 3 of the algorithm for the various test signal frequencies. Also shown are the theoretical curves for various values of σ_Δ which are prediced by equation (7). The plots show fairly consistent results, indicating that the quadratic behavior of C_2 as a function of ω_o is being observed.

The actual estimates of σ_n and σ_Δ are shown in Figures 4 and 5. Measured values of σ_n range consistently from 0.4 to 0.5 over the entire measurement band. These results are only slightly worse than those predicted for an ideal 8-bit converter, indicating a relatively low-noise converter. Sample time jitter measurements show sample time standard deviations on the order of 3 to 4 psec over the entire measurement band.

V. Conclusions

A straightforward test has been developed which allows for the measurement of random sample-time jitter in ADCs. The test is based on driving the converter using a high-frequency sinusoidal test signal. Simulation and experimental results have shown the test may provide accuracy on the order of 1 psec.

References

[1] F.H. Irons, D.M. Hummels, and I.N. Papantonopoulos, "ADC dynamic error modeling," Submitted to *IEEE International Symposium on Circuits and Systems*, ISCAS 95, Seattle Wa, May 1995.

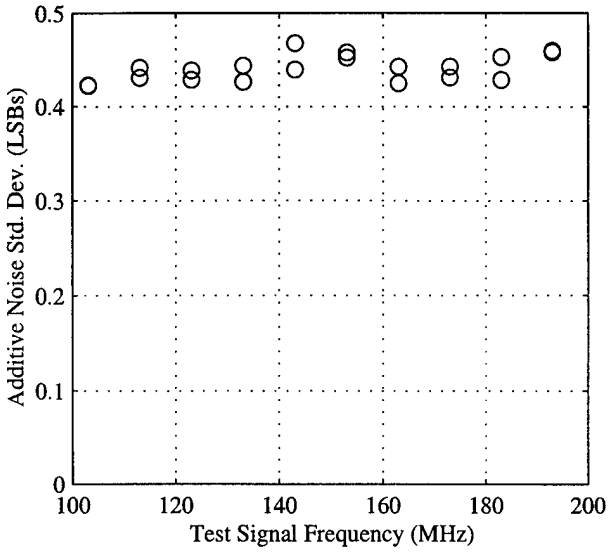

Figure 4: Measurement of σ_n for a Tek AD-20.

[2] F.H. Irons, D.M. Hummels, and S.P. Kennedy, "Improved compensation for analog-to-digital converters" *IEEE Trans. CAS*, pp. 958-961, August 1991.

[3] J.R. Gray and S.C. Kistopoulas, "A precision sample and hold circuit with sub-nanosecond switching," *IEEE Trans. Circuit Theory*, pp. 389-396, Sept. 1964.

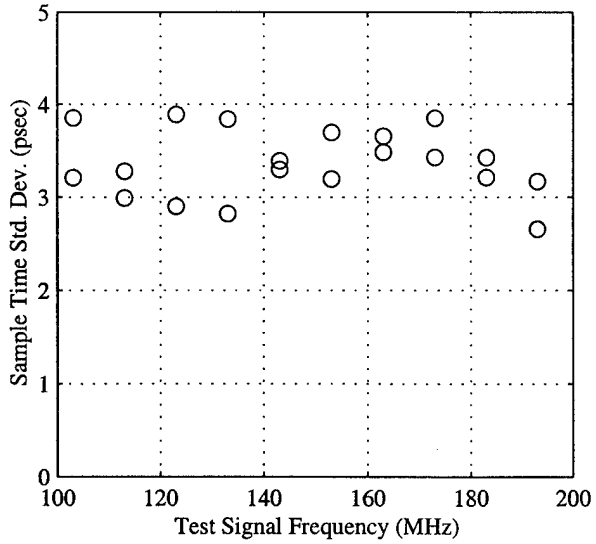

Figure 5: Measurement of σ_Δ for a Tek AD-20.

A Fast Orthogonal Search Algorithm For Radial Basis Function Neural Networks

W. Ahmed, D. M. Hummels and M. T. Musavi
Department of Electrical and Computer Engineering
University of Maine, Orono, Maine 04469

Abstract— **This paper presents a fast orthogonalization process to train a Radial Basis Function (RBF) neural network. The traditional methods for configuring the RBF weights is to use some matrix inversion or iterative process. These traditional approaches are either time consuming or computationally expensive, and often do not converge to a solution. The goal of this paper is first to use a fast orthogonalization process to find the nodes of the RBF network which produce the most improvement on a target function, and then to find the weights for these nodes. Several applications of RBF networks using this fast orthogonal search technique has been conducted and a classification problem is presented. The problem involves classification of human chromosomes, which is a highly complicated 30 dimensional and 24 class problem. Experimental results will be presented to show that the fast orthogonal search technique not only outperforms the traditional technique, but it also uses much less time and effort.**

I. Introduction

The growth of neural networks has been heavily influenced by the Radial Basis Function (RBF) neural networks. The application of the RBF network can be found in pattern recognition [1, 2], function approximation [3, 4], signal processing [4, 5], system equalization [6] and more. The two most important parameters of a RBF node, the center and the covariance matrix, have been researched throughly [2, 6]. A major issue handled by these researchers is the reduction of the number of nodes. This reduction involves clustering of the input samples without any consideration of the target function, or the convergence of the weights. The weights (the most significant component of any neural network) of the RBF network were left untouched by most of the researchers. This oversight is not ignorance but a confidence on the traditional approaches. For RBF weights, the traditional approaches only work when the training samples are well behaved. In real life, the training samples are not well behaved causing major problems on finding the RBF weights. The Issue of this paper is to find a set of most significant nodes and their weights for a given network, using a technique which considers both the structure of the input parameter space and the target function to which the network will be trained.

The traditional approach to design an RBF network is to first select a set of network parameters (number of nodes, node centers, node covariances) and then find the weights by formulating the network by solving a least squares (LS) formulations of the problem. Orthogonal decomposition techniques may be used to provide an orthogonal basis set for a LS problems. An orthogonal scheme was used by Chen et. al. [7] in RBF networks to simultaneously configure the structure of the network and the weights. The orthogonal search technique presented by Chen is cumbersome, and requires redundant calculations making it non-suitable for reasonable size networks.

A similar fast orthogonal search technique has also been developed by Korenberg et. al. [8, 9] for nonlinear system identification. This procedure also includes redundant calculations and was highly customized to the problem of finding the kernels for a nonlinear system with random inputs. This paper presents an efficient fast orthogonal search eliminating the redundancy of [7, 8]. The resulting algorithm is directly applicable to RBF networks and a wide variety of other LS approximation problems.

II. Radial Basis Function Neural Network

A. RBF structure

The RBF Neural Network gained its popularity for its simplicity and speed. RBF is a simple feed forward neural network with only one hidden layer, and an output layer. The hidden layer consists of a set of neurons or nodes with radial basis functions as the activation function of the neuron. A Gaussian density function is the most widely used activation function and assumed throughout this paper. The output layer is a summing unit, which adds up all of the weighted output of the hidden layer. Figure 1 illustrates the RBF network.

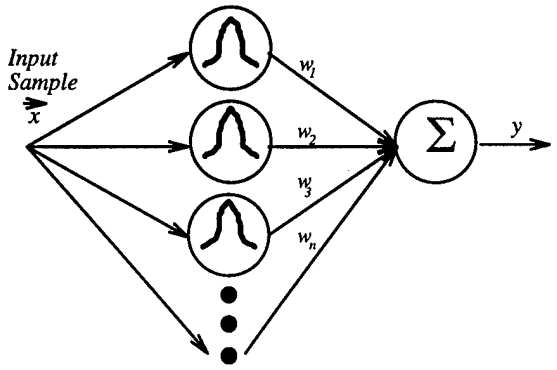

Figure 1: Network structure for the Radial Basis Function Neural Network.

The output of the RBF network is given by

$$\hat{y} = f(\vec{x}) = \sum_{k=1}^{N} w_k \phi_k(\vec{x}), \tag{1}$$

where

$$\phi_k(\vec{x}) = (2\pi)^{-p/2}|\Sigma_k|^{-\frac{1}{2}} e^{-\frac{1}{2}(\vec{x}-\vec{c}_k)\Sigma_k^{-1}(\vec{x}-\vec{c}_k)^T}. \tag{2}$$

Above, N is the number of network nodes, p is the dimensionality of the input space $\vec{x}$, and w_k, $\vec{c}_k$, and Σ_k represent the weight, center, and the covariance matrix associated with each node. In the above equation the output of the network is a scalar quantity for simplicity, but the network can have any number of outputs.

In supervised learning, if $(\vec{x}, y)$ is a input output pair, where $\vec{x}$ is the input and y is the desired output, then the network should learn the mapping function f, where $y = f(\vec{x})$. The training is done using the M training sample pairs $(\vec{x}_1, y_1), (\vec{x}_2, y_2), \cdots (\vec{x}_M, y_M)$. The output vector containing the M outputs of the network can be written using the following matrix form,

$$\hat{\vec{y}} = \Phi\vec{w}, \tag{3}$$

where $\hat{\vec{y}}$ is an M dimensional vector, $\vec{w}$ is the N dimensional weight vector. Each column of the Φ matrix contains the output of a node for all M training samples.

The problem of finding the network weights reduces to finding the vector $\vec{w}$ which makes the network output $\hat{\vec{y}}$ as close as possible to the vector of desired network outputs $\vec{y} = [y_1\ y_2\ \cdots y_M]^T$. Generally, $\vec{w}$ is determined by finding the least square (LS) solution to

$$\Phi\vec{w} = \vec{y}. \tag{4}$$

The method for finding the solution to (4) depends in large part on the structure of the network being designed. One popular scheme is to center a node of the network on each of the input training samples ($\vec{c}_k = \vec{x}_k, k = 1, 2, \cdots, N$). In this case the matrix Φ is square, and the weights are given by $\vec{w} = \Phi^{-1}\vec{y}$ provided that Φ is nonsingular. However, calculation of Φ^{-1} is often problematic, particularly for large networks.

Often, the number of nodes is much less than the number of training samples. In this case the system of equations (4) is overdetermined, and no exact solution exists. Various alternative methods of finding the weights in this case are discussed in the following section.

B. Solving for the Weights

B.1. Orthogonal Search

Another technique for solving the least squares problem is the orthogonal search technique. The orthogonalization of Φ can be found by using the Householder transformation [10], or the Gram-Schimidt orthogonalization procedure [7, 10, 15].

The weights can be found by substituting the orthogonalization of the Φ into equation (??), and using the method of forward substitution, and backward substitution [10].

This procedure does not work when Φ is singular, and it does not give us any insight about the network structure. A more useful approach is to use the orthogonal basis vectors to choose a set of nodes which reduces some error criteria for solving the least squares problem. Chen [7] presents one such algorithm, derived from the Gram-Schimidt procedure, to select the most 'significant' nodes one by one. In [8] a similar algorithm was also presented to select basis function one by one, with emphasis on the characterization of nonlinear systems with random inputs. Korenberg et. al. [8] presents an improved, fast version of orthogonal search technique described in [16].

During each step of Chen's algorithm [7], the following procedure is used to search a set of candidate nodes to determine which node will be added to the currently selected set.

1. For each candidate, find the component of the basis vector associated with that node which is orthogonal to all of the currently selected nodes. Either the Gram-Schimidt or the Householder technique may be used. Call this component $\vec{q}_i$.

2. For each candidate, evaluate the projection of the target vector $\vec{y}$ onto the unit vector in the direction of the orthogonal component

$$p_i = \vec{y}^T \frac{\vec{q}_i}{\|\vec{q}_i\|}. \tag{5}$$

This component gives the length of the change in $\vec{y}$ which will result if the i^{th} vector is added to the currently selected set.

3. Choose the node which gives the largest value of p_i - this is the node which will provide the greatest reduction in the mean-square error.

The importance of choosing the nodes one by one is significant in several aspects. Selection of nodes one by one provides an insight to the approximation problem, and the network structure. This insight can be used to further modify the network. This selection procedure will also let us meet some physical limitations. Nodes mean connections, so a reduction of nodes will provide a corresponding reduction of connections which can be very useful for hardware applications. Finally, this procedure does not involve the selection of an arbitrary thresholding like the SVD method which can affect our solution. The desired number of nodes can be easily chosen just by looking at the error behavior.

The orthogonal search method is very useful, but not so practical. The computational complexity of the procedure is not generally practical for networks of reasonable size. However, the above algorithm may be shown to be extremely redundant. In section IIIan algorithm will be presented to implement the orthogonal search technique efficiently. The algorithm will perform the same orthogonalization procedure without explicitly calculating the orthogonal set.

III. The Fast Orthogonal Search

In this section a simple, fast algorithm will be developed to find a set of weights that are best for the given network. The technique will suggest the number of nodes needed by the network elliminating redundant nodes. The procedure may be shown to be a computationally efficient procedure for implementing the orthogonal search technique of section B.1. Similar fast orthogonal search techniques have been used by Korenberg [8, 9] for time-series analysis, system identification, and signal identification problems.

A. The Fast Orthogonal Search Algorithm

The problem of solving for the RBF weights from the equation (3) is the issue of this section. Like the orthogonal search technique, during each iteration of the algorithm a set of candidate nodes will be considered to identify which node will provide the best improvement to the approximation of $\vec{y}$. This node will be added to the network, and the procedure continues until either an error criterion is met or the number of nodes in the network reaches a desired value. Unlike the orthogonal search technique, the orthogonal basis set associated with the selected set of nodes is never explicitly calculated, significantly reducing the computational burden of the procedure.

The following algorithm presents the appropriate steps to implement the technique.

1. Store all the node outputs in the set $\{\vec{\phi}_j\}$. and initialize the following variables:

$$\begin{aligned} \alpha_i &= \vec{\phi}_i^T \vec{y}, \\ \xi_i^2 &= \vec{\phi}_i^T \vec{\phi}_i, \\ x_i &= [0 \times 1 \ \ vector], \\ U &= [0 \times 0 \ \ matrix], \end{aligned} \tag{6}$$

here, $i = 1, 2, \cdots, N$. Also set

$$Error = \vec{y}^T \vec{y},$$
$$Number\_node\_selected = 0.$$

2. The iteration begins here.
Find the maximum value of α_i^2/ξ_i^2 for all i. Let's say the maximum is at $i = k$.

3. Set,

$$\tilde{U} = \begin{bmatrix} U & \vec{x}_k \\ 0 & \xi_k \end{bmatrix}. \tag{7}$$

4. Update the $\vec{x}_i$ as in equation (??), by finding β_i first,

$$\beta_i = \frac{1}{\xi_k}(\vec{\phi}_k^T \vec{\phi}_i - \vec{x}_k^T \vec{x}_i) \tag{8}$$

giving,

$$\tilde{\vec{x}}_i = \begin{bmatrix} \vec{x}_i \\ \beta_i \end{bmatrix}. \tag{9}$$

5. Update ξ_i as in equation (??),

$$\tilde{\xi}_i^2 = \xi_i^2 - \beta_i^2. \tag{10}$$

6. Finally update α as in equation (??) by,

$$\tilde{\alpha}_i = \alpha_i - \frac{\alpha_k \beta_i}{\xi_k}. \tag{11}$$

In the above, from step 4 to step 6 updating is only required when $i \neq k$.

7. Keep repeating from step 4 through step 6 until all the nodes in the set $\{\vec{\phi}_j\}$ have been updated.

8. Delete the k^{th} node from the set $\{\vec{\phi}_j\}$.

9. Increment the *Number_node_selected* by one, and set $Error = Error - \alpha_k^2/\xi_k^2$. If *Error* is less than some error threshold, or the *Number_node_selected* is equal to the desired number of nodes then go to step 10, otherwise go through the steps 2 - 8 again.

10. At this stage we have a U matrix giving the Cholesky decomposition of Φ

$$\Phi^T \Phi = U^T U. \tag{12}$$

Note that here Φ does not contain the response from all nodes as in section ??, Φ is formed only from the set of nodes that reduce the sum squared error of $\hat{\vec{y}}$. We can use the equation above in the normal equation (??), and then solve for the weights of the selected nodes by the method of forward substitution, and backward substitution.

The algorithm presented in this section not only implements the technique to find the weights of a given network, but also allows the user to select the number of nodes. This selection has physical significance since there may exist hardware or software limitations on implementing nodes. The algorithm finds the best fixed number of nodes rather than just an arbitrary choice of nodes. If the number of nodes is not an issue than one may be able to find a better network by choosing a sum square error threshold. Also we can look at how the error is behaving as the nodes have been added to the network. This provides an indication of whether addition of a node really makes a difference or not.

IV. A pattern Recognition Application

The application of RBF neural network is widespread. RBF networks have been successfully applied in pattern recognition [1, 2], function approximation [4], time series prediction [4], signal detection [5, 17] and many other important problems.

A complicated and challenging application of RBF will be discussed in this section. "Karyotyping", the classification of the chromosome in a metaphase into the 24 normal classes has been a very important issue in the medical field for many many years. The automation of karyotyping by computers has been in development for about 25 years [21]. Classification of chromosomes involves finding a good set of features to describe a chromosome, and a classification technique to identify the chromosomes using the features. The RBF neural network developed in this report will be used for the classification of chromosome given a set of features.

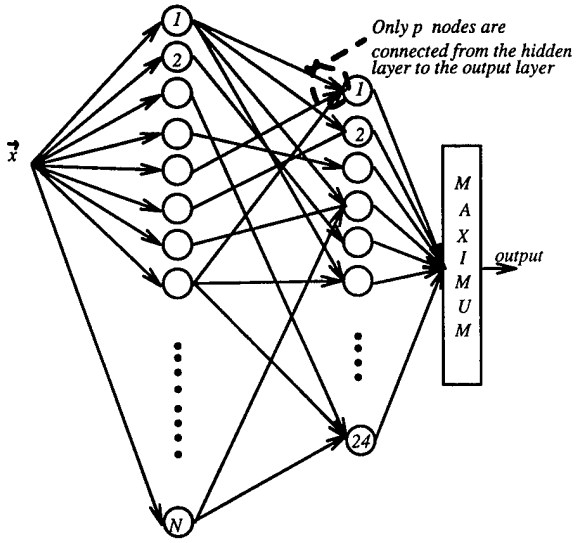

Figure 2: Network structure for the RBF Network for Chromosome Classification.

A. RBF Network for Chromosome Classification

The problem of karyotyping involves classifying the chromosome of 30 features (for the data base used here) into 24 different classes. The chromosomes in a cell consists of 22 pairs of autosomes, one of each pair inherited of each parent, and two sex chromosomes (an X and Y for male, and two X's for female). Classification of a cell correctly requires classification of all 24 classes of a cell. Rather than classifying a cell, this report will look at the classification per class. So here the problem is to only classify each pair of autosomes, and the sex chromosomes.

The RBF structure for the Chromosome classification is slightly different than the one given in Figure 1 of section II. The output layer of the network consists of 24 output nodes to provide a probability measure for each of the 24 classes. The decision of the network is the node that gives the highest output. One major advantage of the fast orthogonal search technique will be evident here. Figure 2 illustrates the RBF network for chromosome classification. For the standard RBF networks, all the nodes of the hidden layer are connected to all the nodes of the output layer. The fast orthogonal search will be able to reduce the insignificant connections. The reduction of the nodes can be of very large scale for a multiclass problem like the chromosome classification. The following result and analysis section will show this phenomenon.

B. Results and Analysis

The Chromosome database used for evaluating the method is the Copenhagen database [20, 21, 22]. Each

pattern of this database is an autosome, the X sex chromosome, or the Y sex chromosome. Each pattern consists of a set of 30 different features, which are the measurements of the normalized area, size, density, normalized convex hull perimeter, normalized length, area, centromeric index, mass centromeric index, length centromeric index, the weighted density distribution density, and others [20].

The RBF neural network was trained with 1000 training patterns. The initial nodes of the network were placed on first 500 of these 1000 patterns. The covariance of each node was chosen to be diagonal with initial diagonal elements equal to the estimated variance of the class the node belongs to, that is

$$\sigma_{ik}^2 = E\left\{(x_{jk} - c_{ik})^2\right\} \qquad \vec{x}_j, \vec{c}_i \in \{class\ l\}. \quad (13)$$

Where σ_{ik}^2 denotes the k^{th} diagonal element of the covariance matrix Σ_i for the i^{th} node, and c_{ik} is the k^{th} component of the i^{th} node center, and x_{jk} is the k^{th} feature of the pattern $\vec{x}_j$. The diagonal elements of the covariance matrix were adjusted to separate the closest discriminating nodes. The adjustment was made by multiplying the diagonal element of the covariance matrix by a fraction of the distance between two overlapping classes, that is

$$\Sigma_i = \gamma\Sigma_i(\vec{c}_j - \vec{c}_i)^T\Sigma_i^{-1}(\vec{c}_j - \vec{c}_i), \qquad \vec{c}_j, \vec{c}_i \notin \{class\ l\}. \quad (14)$$

Where γ is a constant less than 1.0. The fast orthogonal search was used to only find the best 40 nodes per class and their weights. The search technique successfully found the desired nodes. Figure 3 shows the training error for class 1 as each node was added in.

Notice here that only 40 out of 500 RBF nodes are connected to each output node. By looking at figure 3, we see that the training error for class 1 has leveled off by the introduction of the 40^{th} node, implying that introducing another node may not improve the performance at all. Similar conclusion can be made for training other classes.

After training the network, the network was tested by a test set of 3000 patterns different from the training set. The percent error for this test set was 3.84%, which was slightly lower than the ones presented by [20, 21, 22].

The number of initial nodes from which to select a set of nodes of the network can influence the performance. Figure 4 illustrates the results of some simple tests where the number of initial node assignment was changed for the training procedure described above. The plots in figure 4 gives the percent error (y-axis) for a network constructed by selecting a set of nodes per class from a larger set of initial nodes (x-axis). The test was performed to select 10, 20, 30, 40, 50 and 60 nodes per class from the initial sets. The percentage error was generally lower than the error given by [21, 22]. From Figure 4 we can also see that the percent error for 10 selected nodes per class is

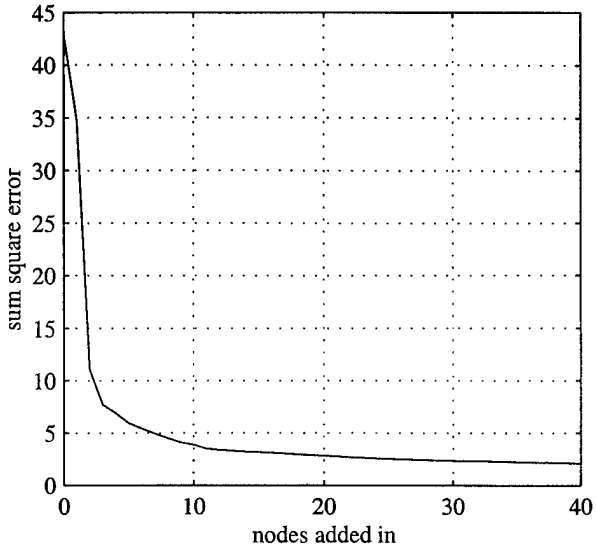

Figure 3: The sum squared error for training class 1 of the Chromosome problem.

higher than all other selections. This suggests that 10 nodes per class is not necessarily the best. On the other hand, selection of 60 nodes does not improve the percent error at all suggesting overtraining. The smaller initial set of (100-300) nodes are good, but not the best (too little to choose from) and the percent error increases as the initial nodes increases (too many to choose fro). One other important issue is that even though the percent error is changing due to the selection process, the difference between the best performance and the worst performance of all the test is merely 1.82%, suggesting that the fast orthogonal search procedure is finding the best possible selection for a given network.

V. Conclusion

This paper presented an approach to configure the most significant component of the RBF neural networks, the weights.

The method provides a simple way to find the most significant nodes of the network and their weights. The technique of fast orthogonal search is implementable using a simple 10 step algorithm. Traditional approaches require significantly more computations. The technique provides a solution regardless of the network parameters. The provided solution is the best to match the target function. The traditional approaches may not converge, or may produce an erroneous solution. The solution is correlated with the target function, so scaling of any node will not affect the network output. This is in contrast to many of the traditional approaches, where node activation func-

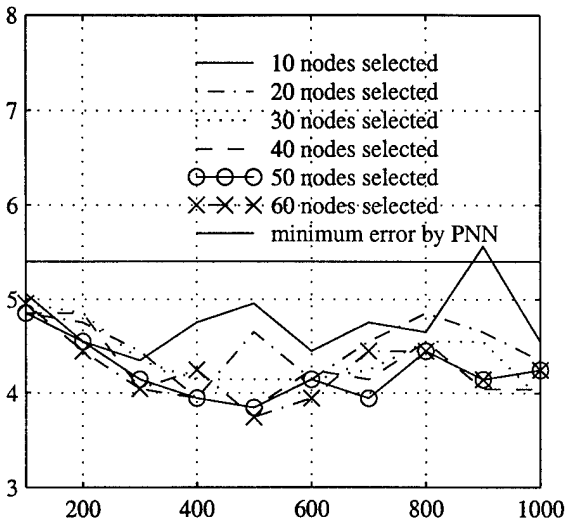

Figure 4: The percent error for the network for different number of the initial node selection

tions must be carefully normalized. The approach gives a clear indication of the number of nodes to be used. Nodes should be added only until addition of a node does not improve the output significantly. Several important applications of the RBF network have been tried out and two applications are provided in this paper. The result shows that the orthogonal search technique gives better performance than that of the other approaches.

The fast orthogonal search technique is a significant improvement over existing RBF training techniques. The method provides good insight into the concerned problem, and the size of the network required to address the problem.

References

[1] J. Moody, C.J. Darken, "Fast Learning in Networks of Locally Tuned Processing Units," *Neural Computation*, vol. 1, pp. 281-294, 1989.

[2] M.T. Musavi, W. Ahmed, K.H. Chan, K.B. Faris, D.M. Hummels, "On the Training Algorithm for Radial Basis Function Classifiers," *Neural Networks*, vol. 5, pp. 595-603, 1992.

[3] D.S. Broomhead, D. Lowe, "Multivariable Functional Interpolation and Adaptive Networks," *Complex Systems*, vol. 2, pp. 321-355, 1988.

[4] R. D. Jones, Y. C. Lee, W. Barnes, G. W. Flake, K. Lee, P. S. Lewis, S. Qian, "Function approximation and Time Series Prediction with Neural Networks," *Proc. of the IJCNN*, pp 649-665, June 1990.

[5] W. Ahmed, D.M. Hummels, M.T. Musavi, "Adaptive RBF Neural Detection in Signal Detection," *Proceedings of International Symposium on Circuits and Systems (ISCAS '94)*, pp 265-268, May 29 - June 2, 1994, London, U.K.

[6] S. Chen, B. Mulgrew, P.M. Grant, "A clustering technique for Digital Communications Channel Equalization Using Radial Basis Function Networks" *IEEE Transactions on Neural Networks*, vol. 4, No. 4, pp. 570-579, Mar. 1993.

[7] S. Chen, C.F.N. Cowan, P.M. Grant, "Orthogonal Least Squares Learning Algorithm for Radial Basis Function Networks" *IEEE Transactions on Neural Networks*, vol. 2, No. 2, pp. 302-309, Mar. 1991.

[8] M.J. Korenberg, L.D. Paarmann, "Orthogonal Approaches to Time-Series Analysis and System Identification," *IEEE SP Magazine*, July 1991, pp. 29-43.

[9] L.D. Paarmann, M.J. Korenberg, "Accurate ARMA Signal Identification - Empirical Results," *Proceedings of Midwest Symposium*, pp 110-113,1991.

[10] L.L. Scharf, *Statistical Signal Processing, Detection, Estimation, and Time Series Analysis*, Addision-Wesley Inc., 1991.

[11] N. Weiner, *Nonlinear Problems in Random Theory*, The Technology Press of MIT and John Wiley & Sons, Inc., New York, 1958.

[12] A.A. Desrochers, "On An Improved Model Reduction Technique for Nonlinear Systems" *Automatica*, vol. 17, pp. 407-409, 1981.

[13] D.M. Hummels, W. Ahmed, M.T. Musavi, "Adaptive Detection of Small Sinusoidal Signals in Non-Gaussian Noise using a RBF Neural Network," to be published in 1994 in *IEEE Transactions of Neural Networks.*

[14] J. Piper, E. Granum, "On Fully Automatic Feature Measurement for Banded Chromosome Classification", *Cytometry* vol. 10 pp. 242-255, 1989.

[15] W. P. Sweeney Jr., M.T. Musavi, J.N. Guidi "Classification of Chromosomes Using Probabilistic Neural Network", *Cytometry* vol. 16 pp. 17-24, 1994.

[16] W. P. Sweeney Jr. *Classification of Human Chromosomes Using Probabilistic Neural Network*, M.S Thesis, University of Maine, Orono, Maine, August 1993.